UNIVERSIDADE ESTADUAL DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
MESTRADO EM COMPUTAÇÃO APLICADA

JONAS ORSO

Comparativo do Impacto do Idioma e da Modelagem de Dados no Desempenho de Modelos de Linguagem na Tarefa de Text-to-SQL

JOINVILLE
2025

JONAS ORSO

Comparativo do Impacto do Idioma e da Modelagem de Dados no Desempenho de Modelos de Linguagem na Tarefa de Text-to-SQL

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do grau de Mestre em Computação Aplicada

Orientador(a): Prof. Dr. André Tavares da Silva

JOINVILLE

2025

Ficha catalográfica elaborada pelo programa de geração automática da Biblioteca Universitária Udesc, com os dados fornecidos pelo(a) autor(a)

Orso, Jonas
Comparativo do Impacto do Idioma e da Modelagem de Dados no Desempenho de Modelos de Linguagem na Tarefa de Text-to-SQL / Jonas Orso. -- 2025. 114 p.

Orientador: André Tavares da Silva
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação , Joinville, 2025.

1. Text-To-SQL. 2. LLMS. 3. Zero-Shot. 4. Esquema de Banco de Dados. I. Tavares da Silva, André. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação . III. Título.

AGRADECIMENTOS

Agradeço, em primeiro lugar, ao meu orientador, pela confiança, pela exigência intelectual e pela atenção constante ao longo de todo o percurso desta pesquisa. Sua leitura cuidadosa, as críticas precisas e o estímulo à autonomia foram decisivos para que este trabalho alcançasse a maturidade necessária. Estendo o agradecimento aos professores do Programa, cujas disciplinas, seminários e conversas informais ampliaram meus horizontes teóricos e metodológicos, e à banca examinadora, pela generosidade da leitura e pelas contribuições que aprimoraram a versão final deste texto.

Registro meu reconhecimento à Universidade do Estado de Santa Catarina, em especial ao Programa de Pós-Graduação, pelo ambiente acadêmico qualificado e pelo suporte institucional. Agradeço, de modo particular, pela concessão da bolsa, que viabilizou a dedicação necessária e deu condições para a realização dos experimentos e para a escrita desta dissertação.

Agradeço, com carinho e respeito, à minha família. Aos meus familiares, pelo incentivo silencioso, pelas mensagens de ânimo e pela compreensão nos momentos de ausência. À minha esposa e às minhas filhas, pela paciência diante das longas horas de estudo, pela compreensão das rotinas alteradas e pelo afeto que me sustentou quando o cansaço parecia maior do que o avanço. Este trabalho também é de vocês, que me lembraram, nos dias bons e nos dias difíceis, do sentido de seguir em frente. Agradeço, também, a Deus por ter nos sustentado até aqui.

RESUMO

O acesso efetivo a dados públicos constitui condição fundamental para a promoção da transparência e da inovação; contudo, uma parcela significativa dos usuários ainda enfrenta dificuldades para converter questões cotidianas em consultas e informações capazes de gerar valor a partir desses dados. Registros administrativos, planilhas e documentos dispersos podem ser normalizados e consolidados em bases relacionais, tornando o conteúdo consultável de forma consistente; a partir desse ponto, a mediação por modelos *Text-To-SQL* permite um acesso conversacional, no qual o usuário formula a demanda em linguagem natural e o sistema produz a consulta SQL correspondente, sendo assim, este trabalho avalia, em cenário zero-shot, o desempenho de modelos de linguagem de grande escala na geração de consultas SQL a partir de linguagem natural aplicada a dados públicos de Santa Catarina. Construiu-se uma base bilíngue em SQLite com quatro variantes que combinam tabela única e esquema estrela, em português e inglês, e um conjunto de cinquenta perguntas com SQL de referência traduzidas para ambos os idiomas, totalizando duzentos pares. A inferência foi realizada localmente com modelos abertos e um único prompt prescritivo que exige a produção de um único comando válido, adotando-se uma métrica de acurácia composta que considera simultaneamente a equivalência de resultados em execução e a correspondência semântica e estrutural. Os resultados mostram vantagem consistente da tabela única sobre o esquema estrela e efeito favorável, ainda que não absoluto, do alinhamento entre o idioma do enunciado e o do esquema. Entre os modelos avaliados, o Qwen apresentou o melhor desempenho global, com de acurácia de 71,74 por cento na tabela única em português e de 60,00 por cento no esquema estrela em cenário cruzado, superando as demais opções em média nas diferentes combinações. Como contribuições, o estudo disponibiliza um recurso experimental bilíngue reprodutível, um protocolo de avaliação compatível com execução local e evidências sobre o impacto do desenho do banco e do idioma no *Text-To-SQL* em português.

Palavras-chave: *Text-To-SQL*; LLMS; Zero-Shot; Esquema de Banco de Dados.

ABSTRACT

Effective access to public data is a prerequisite for transparency and innovation, yet many users face barriers to turning everyday questions into actionable results. Administrative records, spreadsheets, and scattered documents can be normalized and consolidated into relational databases, making the content consistently queryable; from that point, mediation via *Text-To-SQL* models enables conversational access, whereby the user formulates a request in natural language and the system produces the corresponding SQL query, reducing cognitive load and expanding the social reuse of information. Accordingly, this study evaluates, in a zero-shot setting, the performance of large language models in generating SQL from natural language over public data from Santa Catarina. We constructed a bilingual SQLite corpus with four variants that combine single-table and star-schema designs in Portuguese and English, and a set of fifty questions with reference SQL translated into both languages, totaling two hundred pairs. Inference was run locally with open models and a single prescriptive prompt that requires the production of one valid command, and we adopted a composite accuracy metric that jointly considers execution-level result equivalence and semantic and structural correspondence. The results show a consistent advantage for the single-table design over the star schema and a favorable, albeit not absolute, effect of aligning the language of the prompt with that of the schema. Among the evaluated models, Qwen achieved the best overall performance, with peak accuracies of 71.74 percent on the single-table design in Portuguese and 60.00 percent on the star schema in a cross-lingual setting, outperforming the alternatives on average across the different combinations. As contributions, the study releases a reproducible bilingual experimental resource, an evaluation protocol compatible with local execution, and evidence on the impact of database design and language on *Text-To-SQL* in Portuguese.

Keywords: *Text-To-SQL*; LLMS; Zero-Shot; Database Schema.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATIS	*Air Travel Information System*
BERT	*Bidirectional Encoder Representations from Transformers*
DBA	*Database Administrator*
DML	*Data Manipulation Language*
GAP	*Generation-Augmented Pre-training*
GPT	*Generative Pretrained Transformer*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
ILP	*Inductive Logic Programming*
LLMs	*Large Language Models*
LSTM	*Long Short-Term Memory*
MBAs	Mecanismo de Busca Acadêmico
MSL	*Maximum Sequence Length*
NLP	*Natural Language Processing*
OLAP	*Online Analytical Processing*
PEFT	*Parameter-Efficient Fine-Tuning*
QA	*Question Answering*
RNA	Redes Neurais Artificiais
RNN	*Recurrent Neural Networks*
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Languagem*
SVMs	*Support Vector Machines*
TI	Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

A intensificação da digitalização de sistemas e serviços produziu um crescimento expressivo no volume de dados mantidos em bancos relacionais, ao mesmo tempo em que consolidou a linguagem SQL como instrumento central de consulta e manipulação de estruturas tabulares. Embora amplamente difundida, a proficiência em SQL requer formação técnica específica, o que restringe o acesso direto à informação por parte de profissionais de áreas não técnicas e de usuários leigos. Essa barreira cognitiva e procedimental torna-se particularmente evidente quando se pretende explorar bases volumosas, heterogêneas e sujeitas a convenções semânticas próprias de cada domínio.

Nesse cenário, a conversão de linguagem natural em SQL, usualmente denominada *Text-To-SQL*, constitui um passo relevante na democratização do acesso a dados estruturados, pois permite que perguntas formuladas em linguagem humana sejam traduzidas automaticamente em consultas executáveis. A ideia, presente desde os sistemas pioneiros de compreensão de linguagem natural orientados a bases de dados, remonta a LUNAR, voltado à análise de amostras lunares (WOODS, 1972), a CHAT-80, que explorava raciocínio lógico com PROLOG sobre uma base enciclopédica (WARREN; PEREIRA, 1980), a iniciativas de padronização de corpus como o ATIS, no domínio de viagens aéreas (DAHL *et al.*, 1994), e ao GeoQuery, que consolidou o uso de aprendizado supervisionado para consultas geográficas (ZELLE; MOONEY, 1996). O advento do aprendizado profundo reconfigurou o campo com benchmarks como WikiSQL e Spider, que estimularam arquiteturas neurais de tradução semântica, a exemplo de Seq2SQL (ZHONG; XIONG; SOCHER, 2017), SQLNet (XU; BAO; XU, 2017) e IRNet (GUO *et al.*, 2019). Apesar desses avanços, a ênfase histórica no inglês limita a generalização a outros contextos linguísticos e sociotécnicos, como o português brasileiro, com impactos práticos na adoção em cenários reais.

No Brasil, o acesso à informação pública é assegurado pela Lei nº 12.527, de 2011, que estabelece a transparência ativa e passiva e fundamenta a disponibilização de dados abertos. A existência de portais, catálogos e interfaces de programação não elimina, entretanto, as exigências técnicas inerentes ao uso efetivo dessas fontes, que incluem a compreensão de taxonomias e classificações orçamentárias, a navegação por dicionários de dados, a execução de junções entre tabelas e a superação de

limitações operacionais de APIs e formatos. Em Santa Catarina, como no restante do país, os conjuntos de dados orçamentários obedecem a esse marco jurídico, mas a sua exploração direta por cidadãos, jornalistas e gestores sem suporte técnico especializado permanece desafiadora. Essa conjuntura torna o problema *Text-To-SQL* especialmente relevante para políticas de transparência e para o fortalecimento do controle social.

Este trabalho investiga o desempenho de modelos de linguagem de grande escala na geração de SQL em português, com ênfase na avaliação multilíngue e multiestrutura. Para isso, foi construído um conjunto de dados bilíngue, em português e inglês, composto por informações orçamentárias públicas do Estado de Santa Catarina, de modo a refletir nomenclaturas, códigos e práticas do domínio. A base foi disponibilizada no repositório online github[1] em duas modelagens complementares, uma em esquema estrela e outra em tabela única, com o objetivo de mensurar o efeito do desenho do banco na acurácia das consultas geradas e, simultaneamente, avaliar a sensibilidade dos modelos à correspondência linguística entre o idioma do esquema e o idioma das perguntas.

A avaliação contempla cinco modelos de código aberto representativos do estado da prática recente, a saber, Gemma, OpenChat, Gaia, DeepSeek e Qwen. A seleção considera, de forma pragmática, a disponibilidade de versões instrucionais, a viabilidade de execução em infraestrutura local e o suporte a cenários multilíngues, todos fatores relevantes para adoção em contextos públicos brasileiros. Os experimentos são conduzidos em regime zero-shot, sem ajuste fino, a fim de aferir a capacidade de generalização nativa dos modelos frente à tradução semântica entre linguagem natural e SQL.

A pesquisa está orientada por questões que articulam idioma e modelagem de dados. Investiga-se se o desempenho dos modelos varia quando o idioma do esquema e o idioma das perguntas estão alinhados ou desalinhados e se a organização dos dados em esquema estrela ou em tabela única influencia a precisão de execução e a consistência sintática das consultas geradas. Ao responder a essas questões, o estudo pretende oferecer evidências comparativas que auxiliem pesquisadores e gestores públicos na definição de estratégias de publicação e exploração de dados que privilegiem acessibilidade e qualidade de resposta

[1] Link de acesso á base de dados https://github.com/Jonasorso/Despesas_Sc_Text-To-Sql

1.1 OBJETIVO

O presente trabalho tem como objetivo analisar o desempenho de modelos de linguagem de grande escala na tarefa de transformação de linguagem natural em consultas SQL, considerando variações de idioma e estrutura de banco de dados. A pesquisa busca avaliar o desempenho de modelos de linguagem de grande porte na geração de consultas SQL em diferentes contextos linguísticos, avaliando sua aplicabilidade em bases de dados governamentais brasileiras.

Os Objetivos Específicos (OE) incluem:

- Desenvolver uma base de dados bilíngue composta por informações orçamentárias públicas, estruturada em dois formatos distintos (esquema estrela e tabela única), adequada para experimentos de *Text-To-SQL*;
- Avaliar comparativamente o desempenho de diferentes modelos de linguagem (Gemma, OpenChat, Gaia, DeepSeek e Qwen) em cenários com variação de idioma entre a pergunta e o *schema* do banco de dados aplicando técnicas de zero-shot;
- Investigar a influência da estrutura do banco e da correspondência linguística sobre a acurácia das consultas geradas;

1.2 ESCOPO

O estudo centra-se em dados orçamentários públicos do Estado de Santa Catarina e utiliza um conjunto bilíngue, com esquemas e consultas em português e em inglês, de modo a examinar simultaneamente os efeitos do idioma e do desenho do banco de dados na qualidade das respostas. Para isolar esses fatores, a base é disponibilizada em duas modelagens complementares, um esquema estrela e uma tabela única, permitindo mensurar o impacto da organização dos dados na geração de consultas.

A pesquisa adota exclusivamente o regime zero-shot, sem qualquer etapa de ajuste fino ou treinamento adicional, e emprega modelos abertos representativos do estado da prática recente, selecionados pela viabilidade de execução local e suporte a cenários multilíngues. O objetivo é observar a capacidade de generalização nativa desses modelos diante da tarefa de tradução semântica entre linguagem natural e SQL em um domínio público concreto. As consultas consideradas abrangem

diferentes tipos de categorias, passando por itens mais simples, como consultas simples e indo até operações com junções de tabelas, preservando foco em bancos relacionais padronizados e evitando cenários artificiais com múltiplos bancos heterogêneos ou estruturas incomuns que fujam ao escopo de aplicação prática deste estudo.

1.3 ORGANIZAÇÃO DO TEXTO

Esta dissertação está estruturada em quatro capítulos, além das seções introdutórias, conclusão e dos apêndices.

O Capítulo 2 aborda os fundamentos teóricos que sustentam a pesquisa, contemplando conceitos relacionados à linguagem SQL, aos LLMs (Large Language Models) e às abordagens de conversão de linguagem natural em consultas estruturadas (*Text-To-SQL*). Também são discutidos os principais estudos e *benchmarks* que compõem o estado da arte nessa área, com destaque para os desafios linguísticos e estruturais que motivam a presente investigação.

O Capítulo 3 apresenta os trabalhos relacionados, discutindo pesquisas e iniciativas relevantes sobre geração de SQL a partir de linguagem natural. São analisados os principais *benchmarks*, como o Spider e o WikiSQL, bem como estudos recentes que utilizam LLMs nessa tarefa. O capítulo também destaca as lacunas existentes nas pesquisas em língua portuguesa e as oportunidades que motivaram o desenvolvimento deste trabalho.

O Capítulo 4 descreve o desenvolvimento do trabalho e evidencia o que foi construído. Detalha-se a criação do conjunto de dados orçamentários públicos do Estado de Santa Catarina, o processo de coleta e padronização, a modelagem em duas variantes complementares, esquema estrela e tabela única, e a produção das versões bilíngues do esquema e dos rótulos. Apresenta-se a elaboração do conjunto de consultas em linguagem natural com seus respectivos SQL de referência.

O Capítulo 5 são descritos ainda o *pipeline* de avaliação em cenário zero-shot, a infraestrutura de execução e os artefatos gerados, como *scripts*, versões consolidadas da base e o repositório do material experimental. Esse capítulo consolida as decisões de projeto e as medidas de reprodutibilidade adotadas, demonstrando os resultados do processo de construção da base e do conjunto de questões que sustentam os testes. Neste capítulo também são apresentados os

resultados empíricos obtidos a partir dos experimentos comparativos com modelos de linguagem abertos, conduzidos sobre as duas modelagens de dados e nas combinações de idioma entre esquema e consultas. Relata-se a acurácia composta adotada como métrica, analisa-se o comportamento dos modelos nas diferentes configurações e discutem-se efeitos de idioma e de desenho do banco na geração de SQL.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção de conceitos fundamentais, serão explorados temas como processamento de linguagem natural, modelos de linguagem, aprendizado de máquina e SQL, que se interconectam para abordar os desafios e avanços na tradução de texto em linguagem natural para comandos estruturados em bancos de dados relacionais.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Gonzalez e Lima (2003) afirmam que o processamento da linguagem natural (PLN) trata computacionalmente os diversos aspectos comunicação humana, como sons, palavras, sentenças e discursos, considerando formatos e referências, estruturas e significados, contextos e usos. Em sentido bem amplo, podemos dizer que o PLN visa fazer o computador se comunicar em linguagem humana, nem sempre necessariamente em todos os níveis de entendimento e/ou geração de sons, palavras, sentenças e discursos.

A estrutura deste campo será separada em tópicos menores a fim de compreender como cada ponto se relacionam ao processamento da linguagem e como os modelos conseguem compreender na teoria o significado das palavras e textos. No entanto, também cabe salientar as diferenças entre a forma como cada modelo de processamento de linguagem trata as palavras, de forma que pode haver distinções entre eles, por exemplo, a diferença entre a análise das palavras em modelos *Transformers* e modelo de redes Neurais como LSTM, ver seção 2.2 sobre aprendizado de máquina.

2.1.1 Análise Morfológica

A análise Morfológica refere-se à natureza da composição das palavras e lida com a estrutura das palavras na linguagem, as quais são compostas das menores unidades de significado, por exemplo: desinência, raiz, radical, afixo, tema e vogal temática (BATES, 1993). Também nesse sentido, para Gonzalez e Lima (2003), a morfologia e a sintaxe tratam da constituição das palavras e dos grupos de palavras que formam os elementos de expressão de uma língua.

Desta forma é compreensível que as palavras são classificadas naturalmente já pelos próprios humanos, de forma que o processamento da linguagem se aproveita destas técnicas para também compreender seus significados. Por exemplo utilizando a palavra “processamento”, temos que radical é o "process", onde o núcleo significativo da palavra, derivado do latim *processus* (ação de avançar, progredir). Esse radical está associado à ideia de seguir um curso ou realizar etapas. E também o sufixo nominal "-amento" que indica o resultado de uma ação ou o processo relacionado à raiz verbal. É comum em palavras da língua portuguesa que designam ações ou processos derivados de verbos.

Por meio desta análise, os modelos de linguagem classificam as palavras, e para isso duas técnicas são muito utilizadas: a *stemming* e a *Lemmatization*. Onde o *stemming* é o processo de se reduzir palavras flexionadas e derivadas à sua raiz (*stem*), geralmente sob a forma de uma palavra escrita (BATES, 1993). Por exemplo, a raiz “corr” dá origem a diversas palavras, como correndo, corrida, corredor etc. Para o processo de *Lemmatization* é redução à forma canônica, que, geralmente, reduz os verbos ao infinitivo e os adjetivos e substantivos à forma masculina singular (Arampatzis, 2000), um exemplo desta técnica aplicado é a palavra “correndo”, que seria transformada em seu verbo no infinitivo “correr”.

Vale aqui ressaltar que os *Transformers*, tecnologia mais empregada atualmente tem um processo diferente do que é utilizado pois tem uma abordagem contextualizada e distribuída de representação lexical, que é diferente do *stemming* e da *Lemmatization*. Enquanto o *stemming* e a lematização buscam simplificar ou normalizar as palavras para uma forma base ou raiz (muitas vezes sem considerar o contexto), os *Transformers* são capazes de gerar representações contextuais de palavras com base em suas posições e contextos nas frases (Vaswani *et al.* 2017).

2.1.2 Análise Sintática

Para Gonzalez e Lima (2003), a análise sintática (*parsing*) é o processo de avaliação das diferentes formas de combinar regras gramaticais, com o objetivo de gerar uma árvore que represente a estrutura sintática de uma sentença. Por exemplo, para a frase "O gato correu atrás do rato", a análise sintática identificaria "O gato" como o sujeito, "correu" como o verbo predicado e "atrás do rato" como o objeto.

No entanto, os modelos *Transformers* não realizam um *parsing* explícito com base na construção de árvores sintáticas, como acontece em abordagens tradicionais de processamento de linguagem natural. Em vez disso, eles utilizam o mecanismo de atenção para aprender relações sintáticas e semânticas de forma implícita (Vaswani *et al.,* 2017). Esse mecanismo permite que o modelo identifique quais palavras estão mais relacionadas entre si dentro de uma frase, sem necessidade de estruturas hierárquicas explícitas.

Por exemplo, no contexto da frase "O gato correu atrás do rato", um *Transformer* seria capaz de capturar a relação entre "correu" e "rato" diretamente por meio de pesos de atenção. Essa abordagem torna os modelos mais adaptáveis e eficientes para uma ampla variedade de tarefas, pois permite aprender diretamente a partir dos dados, sem depender de regras ou estruturas linguísticas pré-definidas. Ainda assim, informações sintáticas podem ser incorporadas aos *Transformers* em tarefas específicas, como no caso de sistemas que combinam *parsing* com *embeddings* contextuais.

2.1.3 Variações linguísticas das palavras

As variações linguísticas segundo (JACQUEMIN; KLAVANS; TZOUKERMANN, 1997) e (ARAMPATZIS, 2000) podem ser classificadas em:

• morfológica, quando processos flexionais ou derivacionais criam palavras diferentes, como em “lobo” e “lobos”;

• lexica, quando diferentes palavras são usadas para representar o mesmo significado, como “bolacha” e “biscoito”;

• sintático-semântica, quando a posição relativa das palavras determina frases com significados diferentes, como “biblioteca da ciência” e “ciência da biblioteca”;

• morfossintática, quando variações morfológicas não impedem a manutenção do significado essencial da frase, podendo ser:

• variações substantivo-substantivo, como resultado/agente em “fixação de nitrogênio” e “fixador de nitrogênio”, ou recipiente/conteúdo em “reservatório de água” e “reserva de água”;

• variações substantivo-verbo, como processo/resultado em “fixação de nitrogênio” e “fixar nitrogênio”; e

• variações substantivo-adjetivo, onde um modificador preposicional é substituído por modificador adjetival, como em “variação do clima” e “variação climática”; e

• semântica, quando diversos significados são possíveis para o mesmo objeto linguístico, como “palmas” e “queda da bolsa”.

Desta forma as variações linguísticas podem impactar a forma como as arquiteturas dos modelos de linguagem interpretam e convertem texto em consultas SQL. Por exemplo, variações morfológicas, como "lobo" e "lobos", pedem que o modelo entenda que são a mesma palavra com pequenas mudanças. Variações lexicas, como "calçado" e "sapato", desafiam o modelo a perceber que são sinônimos e significam a mesma coisa. Diferenças na estrutura da frase, como em "fixação de nitrogênio" e "fixador de nitrogênio" (variações sintático-semânticas e morfossintáticas), exigem que o modelo identifique que a ideia principal continua a mesma, mesmo com palavras em posições diferentes. Por fim, as variações semânticas, como "palmas" (que pode significar aplausos ou planta), fazem o transformador precisar de pistas no texto para escolher o significado certo, garantindo que a consulta SQL reflita exatamente o que o usuário quer dizer.

Dessa forma, pode-se afirmar que a maneira como as frases e os textos são escritos influencia diretamente na obtenção dos resultados esperados, podendo gerar impactos tanto positivos quanto negativos.

2.2 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina é um campo da computação que busca desenvolver sistemas capazes de realizar tarefas que, até então, requeriam inteligência humana. O objetivo do aprendizado de Máquina é a construção de programas que melhorem seu desempenho por meio de exemplos (MITCHELL, 1997). Este campo está dividido em algumas abordagens, das quais o aprendizado supervisionado e não-supervisionado são os mais amplamente utilizados. O aprendizado supervisionado, uma das vertentes mais estudadas, envolve o treinamento de modelos a partir de dados rotulados, nos quais o objetivo é fazer previsões ou classificações com base em padrões aprendidos. Dentro deste contexto, diversos algoritmos e técnicas são explorados, como a regressão, utilizada para prever valores contínuos; a classificação, focada em categorizar dados em classes;

as redes neurais, que simulam a estrutura do cérebro humano para resolver problemas complexos; o processamento de linguagem natural, que lida com a interação entre computadores e linguagem humana; e os *Transformers* principalmente no processamento de linguagem.

Esses paradigmas de aprendizado — supervisionado, não supervisionado e semi-supervisionado — são pilares fundamentais no desenvolvimento da inteligência artificial moderna, sendo aplicados em diversas áreas como saúde, finanças, marketing e muitas outras. Enquanto o aprendizado supervisionado utiliza dados rotulados para treinar modelos e o aprendizado não supervisionado identifica padrões em dados não rotulados, o aprendizado semi-supervisionado combina os dois, aproveitando pequenas quantidades de dados rotulados em conjunto com grandes volumes de dados não rotulados para alcançar melhores resultados. Nesta seção, exploraremos as técnicas de aprendizado supervisionado, destacando suas particularidades e aplicações, mas não de forma exaustiva, devido ao fato que esses tópicos serviram de norte e não como objetivo da dissertação.

2.2.1 Aprendizado Supervisionado

No aprendizado de máquina supervisionado, ocorre o treinamento dos modelos a partir de informações previamente rotuladas, ou seja, neste contexto já existe a relação de entrada e saída, que permite a técnica emprega reconhecer o padrão já estabelecido com base nessas informações. Sendo assim, segundo Ludermir (2021), o objetivo do algoritmo é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados. Para rótulos de classe discretos, esse problema é chamado de classificação e para valores contínuos como regressão.

2.2.1.1 Regressão

Modelos de regressão são ferramentas estatísticas amplamente utilizadas para descrever e analisar relações entre variáveis. Esses modelos são costumeiramente utilizados para prever e calcular o valor de uma variável dependente com base em uma ou mais variáveis independentes. A regressão linear, por exemplo,

assume uma relação linear entre as variáveis, enquanto a regressão logística é usada para variáveis dependentes categóricas. Ainda há extensões para dados que não seguem distribuições normais, como os modelos de regressão generalizados, que abrangem diferentes funções de ligação para modelar uma ampla gama de situações. Essas abordagens permitem a interpretação de relações complexas. (PAULA, 2004).

2.2.1.2 Classificação

A tarefa de classificação é amplamente empregada na mineração de dados e consiste em prever a classe de novos exemplos com base em suas características. Para isso, um modelo computacional é desenvolvido utilizando dados de treinamento, que permitem ao classificador identificar relações entre características e classes. Esse modelo é, então, aplicado a um conjunto de teste, cujas classes reais permanecem ocultas ao classificador. Após prever as classes, os resultados são comparados às classes reais para avaliar a acurácia do modelo. Diferentes modelos de classificação podem ser usados, como árvores de decisão, redes neurais, modelos bayesianos e máquinas de vetores de suporte, chamadas de SVMs, do inglês Suport Vector Machine (SILVA, 2011).

2.2.1.3 Redes Neurais Artificiais

As RNAs são modelos computacionais inspirados no funcionamento do cérebro humano. Elas são compostas por unidades simples, chamadas neurônios artificiais, que se conectam para formar redes complexas. Essas redes são capazes de aprender com exemplos, reconhecer padrões e tomar decisões, sendo amplamente utilizadas para resolver problemas complexos, especialmente quando o comportamento dos dados não é completamente conhecido. A criação das RNAs remonta a 1943, com o desenvolvimento de um modelo matemático do neurônio biológico, e evoluiu ao longo das décadas, com avanços significativos a partir de 1982. A escolha da arquitetura ideal para uma RNA é um desafio que envolve experimentação, mas as mais comuns incluem redes com uma ou múltiplas camadas e redes recorrentes. Assim como o cérebro humano, as RNAs aprendem através de

um processo de ajuste de pesos sinápticos, permitindo que elas generalizem informações e façam previsões (FLECK *et al.*, 2016).

2.2.2 Aprendizado Semi-supervisionado

O aprendizado semi-supervisionado combina um conjunto reduzido de exemplos rotulados com um grande volume de dados não rotulados para reduzir o custo de anotação e melhorar a generalização. Estratégias clássicas incluem o auto-treinamento com pseudo-rótulos com vistas complementares, enquanto abordagens contemporâneas impõem consistência a perturbações e usam pseudo-rotulagem de alta confiança; em linguagem natural, a retrotradução é um mecanismo efetivo para gerar pares sintéticos (YAROWSKY, 1995; BLUM*;* MITCHELL, 1998; SENNRICH*;* HADDOW*;* BIRCH, 2016; TARVAINEN*;* VALPOLA, 2017; BERTHELOT *et al.*, 2019). Para *Text-To-SQL*, essas ideias se traduzem na expansão de pequenos conjuntos de pares pergunta–SQL por geração e validação via execução, embora este trabalho avalie modelos exclusivamente em regime *zero-shot*.

2.2.3 Aprendizado Não-supervisionado

No aprendizado não supervisionado, o modelo busca estrutura latente em dados sem rótulos, organizando-os por similaridade, densidade ou geração subjacente. Em processamento de linguagem, a prática que sustenta os modelos atuais é o pré-treinamento auto-supervisionado, usualmente tratado como não supervisionado: modelos causais aprendem a prever o próximo token e modelos mascarados reconstroem lacunas, o que permite induzir regularidades sintáticas e semânticas a partir de grandes corpora sem anotação externa (RADFORD *et al.*, 2018; DEVLIN *et al.*, 2019). Essa etapa é viabilizada pela arquitetura *Transformer*, que modela dependências de longo alcance por mecanismos de atenção e paralelismo, tornando o treinamento escalável (VASWANI *et al.*, 2017). É dessa base que emergem as capacidades zero-shot exploradas nesta dissertação, inclusive na tarefa de *Text-To-SQL*, ainda que o mapeamento entre linguagem natural e consultas estruturadas exija posterior condicionamento ao esquema do banco.

2.2.3.1 Arquitetura Transformer

A arquitetura *Transformers*, proposta por Vaswani *et al.* (2017), introduziu o mecanismo de atenção como operador central para ponderar a relevância relativa de cada *token* dentro de uma sequência, superando limitações de dependências de longo alcance presentes em redes recorrentes. Diferentemente de abordagens sequenciais, os *Transformers* processam o texto em paralelo, o que favorece o treinamento em larga escala. A organização básica é composta por um codificador, responsável por construir representações contextuais da entrada, e por um decodificador, que gera saídas condicionadas a essas representações. Variantes somente decodificador são predominantes nos modelos de linguagem atuais. No pré-treinamento, modelos mascarados realizam modelagem de lacunas e modelos causais realizam modelagem autoregressiva, produzindo representações que se transferem para múltiplas tarefas, entre elas o *Text-To-SQL*, sem necessidade de ajustes manuais extensos.

2.3 LINGUAGEM DE CONSULTA ESTRUTURADA E MODELAGEM DE DADOS

A linguagem de consulta estruturada, do inglês SQL, é uma linguagem de programação projetada para gerenciar e manipular dados em sistemas de gerenciamento de bancos de dados relacionais (SGBDs). Com a capacidade de realizar operações como consulta, inserção, atualização e exclusão de dados, o SQL se tornou uma ferramenta essencial para profissionais de TI e desenvolvedores. Ele permite a comunicação eficiente entre usuários e bancos de dados, proporcionando a organização e análise de grandes volumes de informações de maneira estruturada e acessível. Suas características o tornaram a linguagem padrão para a maioria das aplicações que envolvem bases de dados, sendo amplamente utilizado em áreas como negócios, ciência de dados e desenvolvimento de sistemas.

Já no ponto da modelagem de dados, ela condiciona diretamente a complexidade sintática das consultas, o número de junções necessárias e a clareza semântica do esquema. Estruturas não normalizadas em tabela única tendem a reduzir a profundidade de junções e a simplificar instruções de filtragem e agregação; arranjos dimensionais, como o esquema estrela, equilibram legibilidade e redundância controlada ao centralizar medidas em uma tabela fato e descrever eixos de análise em dimensões; variações mais normalizadas, como o floco de neve, tornam explícitas

hierarquias nas dimensões ao custo de consultas potencialmente mais extensas. Nesta seção, apresentam-se os elementos de SQL pertinentes à formulação de consultas analíticas e, em seguida, discutem-se esses cenários clássicos de organização do catálogo, com ênfase em como as diferentes escolhas estruturais influenciam a redação e a execução de consultas e, por extensão, o mapeamento entre linguagem natural e instruções SQL em tarefas de *Text-To-SQL*.

2.3.1 Linguagem de Consulta Estruturada

De acordo com Elmasri e Navathe (2005), os bancos de dados são componentes fundamentais na sociedade moderna, desempenhando um papel central em diversas atividades cotidianas, como transações bancárias, reservas de hospedagens, consultas bibliográficas, compras online e controle de estoque em supermercados. Essas interações geralmente envolvem pessoas ou sistemas computacionais que acessam bancos de dados para realizar operações e atualizar informações.

A manipulação desses dados é facilitada por uma linguagem padrão amplamente utilizada chamada SQL (*Structured Query Language*). O SQL foi desenvolvido para fornecer uma interface eficiente e padronizada para a interação com bancos de dados relacionais. Por meio dessa linguagem, é possível executar operações essenciais, como a criação e manutenção de estruturas de dados, a inserção, atualização e exclusão de registros, bem como a realização de consultas complexas para recuperar informações armazenadas. O SQL não apenas simplifica o gerenciamento de dados em sistemas de grande escala, mas também garante interoperabilidade entre diferentes plataformas de bancos de dados.

Segundo Elmasri e Navathe (2005), para cada Sistema de Gerenciamento de Banco de Dados (SGBD) existente, é necessário especificar os esquemas conceitual e interno, bem como os mapeamentos entre eles. Para esse fim, utiliza-se a linguagem de definição de dados (Data Definition Language – DDL), que permite ao administrador do banco de dados (Database Administrator – DBA) e aos projetistas definir os esquemas e armazená-los no catálogo do SGBD. Em SGBDs que mantêm a separação entre os níveis conceitual e interno, a DDL é empregada para a definição do esquema conceitual, enquanto a linguagem de definição de armazenamento (Storage Definition Language – SDL) é utilizada para especificar o esquema interno.

Após a definição dos esquemas e o povoamento do banco de dados, os usuários podem manipulá-lo por meio da linguagem de manipulação de dados (Data Manipulation Language – DML), que fornece comandos para operações como recuperação, inserção, remoção e modificação dos dados. Neste trabalho, é dada ênfase às especificações dos comandos relacionados às consultas de busca ou recuperação, uma vez que esse é o foco central da pesquisa.

2.3.2 Comandos de busca em SQL

As consultas de busca realizadas por meio da linguagem de manipulação de dados (DML) são essenciais para recuperar informações armazenadas em um banco de dados. Essas consultas são construídas utilizando o comando *SELECT*, que permite selecionar, filtrar e organizar dados de acordo com critérios específicos. A seguir, são apresentadas as principais cláusulas utilizadas em consultas de busca:

2.3.2.1 Select

A cláusula *SELECT* é a base de qualquer consulta de busca e é utilizada para especificar as colunas que devem ser retornadas no resultado. Por exemplo:

SELECT nome, idade FROM clientes;

Essa consulta retorna os valores das colunas nome e idade da tabela clientes.

2.3.2.2 From

A cláusula *FROM* indica a tabela ou as tabelas de onde os dados serão extraídos. Em consultas mais complexas, ela pode incluir junções (*joins*) para combinar dados de múltiplas tabelas. Exemplo:

SELECT pedidos.id, clientes.nome

FROM pedidos

JOIN clientes ON pedidos.cliente_id = clientes.id;

O comando SQL retorna o ID do pedido e o nome do cliente, unindo as tabelas pedidos e clientes com base na correspondência entre pedidos.cliente_id e clientes.id.

2.3.2.3 Where

A cláusula *WHERE* é utilizada para filtrar os registros que atendem a uma condição específica. Ela permite o uso de operadores lógicos (=, >, <, *LIKE*, *IN*, etc.) para criar critérios de seleção. Exemplo:

```
SELECT nome
FROM clientes
WHERE idade > 30;
```

Essa consulta retorna os nomes dos clientes com idade superior a 30.

2.3.2.4 Group By

A cláusula *GROUP BY* é usada para agrupar registros com base em uma ou mais colunas, sendo frequentemente combinada com funções agregadas, como *SUM*, *COUNT, AVG, MAX* e *MIN*. Exemplo:

```
SELECT categoria, COUNT(*)
FROM produtos
GROUP BY categoria;
```

Essa consulta conta quantos produtos existem em cada categoria.

2.3.2.5 Having

A cláusula *HAVING* é semelhante à cláusula *WHERE*, mas é usada para filtrar grupos de dados após a aplicação do *GROUP BY*. Exemplo:

```
SELECT categoria, COUNT(*)
FROM produtos
GROUP BY categoria
HAVING COUNT(*) > 10;
```

Essa consulta retorna apenas as categorias que possuem mais de 10 produtos.

2.3.2.6 Order By

A cláusula *ORDER BY* organiza os resultados da consulta em ordem ascendente (*ASC*) ou descendente (*DESC*). Exemplo:

```
SELECT nome, idade
FROM clientes
ORDER BY idade DESC;
```

Essa consulta retorna os nomes e idades dos clientes, organizados pela idade em ordem decrescente.

2.3.2.7 Limit

A cláusula *LIMIT* é usada para restringir o número de registros retornados pela consulta, sendo útil para exibir apenas uma parte dos dados. Exemplo:

```
SELECT nome
FROM clientes
LIMIT 5;
```

Essa consulta retorna apenas os cinco primeiros registros da tabela clientes.

2.3.2.8 Combinação de Cláusulas

As cláusulas podem ser combinadas para construir consultas mais complexas. Por exemplo:

```
SELECT categoria, AVG(preco) AS preco_medio
FROM produtos
WHERE estoque > 0
GROUP BY categoria
HAVING AVG(preco) > 50
ORDER BY preco_medio DESC
LIMIT 3;
```

Essa consulta retorna as três categorias com maior preço médio, considerando apenas os produtos com estoque disponível e com preço médio acima de 50.

2.3.3 Junções, chaves e integridade referencial (conceitual)

No modelo relacional, a estrutura lógica do catálogo — relações, atributos e restrições — determina a correção semântica dos dados e a forma canônica das consultas. A formulação original do paradigma relacional estabelece a identificação unívoca de tuplas, a independência lógica e o papel das restrições declarativas como base do projeto de esquemas normalizados (CODD, 1970). Nesse enquadramento, chaves e chaves estrangeiras constituem o mecanismo formal de identificação e de vínculo entre relações, enquanto as junções materializam, na álgebra e no SQL, as correspondências necessárias para combinar dados dispersos em múltiplas tabelas (ABITEBOUL; HULL; VIANU, 1995; ULLMAN, 1988).

A integridade referencial, por sua vez, estabelece que valores de chaves estrangeiras presentes em uma relação devem corresponder a valores existentes de chaves primárias (ou candidatas) na relação referenciada, admitindo políticas explícitas para atualizações e exclusões. A literatura discute condições de segurança para essas estruturas referenciais e os efeitos de operações com nulos e dependências múltiplas, propondo critérios formais para garantir consistência sob modificações no banco (RAFIQ; ORLOWSKA; SADIQ, 1991).

No contexto de geração automática de SQL a partir de linguagem natural, chaves e integridade referencial funcionam como âncoras semânticas para o “caminho de junção” correto entre tabelas, reduzindo ambiguidades na seleção de relações intermediárias e na formulação dos predicados *ON*. A distinção entre junções internas e externas altera a cobertura de registros ausentes, aspecto sensível em consultas agregadas, contagens e totais, o que reforça a importância de restrições declarativas e de esquemas explicitamente normalizados como suporte ao mapeamento entre termos do enunciado e colunas-chave do catálogo (CODD, 1970).

2.3.4 Modelos de Dados

A modelagem de dados para consulta analítica parte de princípios do modelo relacional e dos arranjos lógico-físicos que equilibram integridade, redundância e caminhos de acesso. Em ambientes voltados a leitura, como data *warehouses* e sistemas de apoio à decisão, a organização do catálogo influencia diretamente a complexidade das consultas, a profundidade de junções e a previsibilidade de padrões

de acesso (CHAUDHURI; DAYAL, 1997; ELMASRI; NAVATHE, 2005). Três configurações aparecem recorrentemente na literatura: a estrutura não normalizada em tabela única, o esquema estrela e o esquema floco de neve. Elas diferem no grau de normalização das dimensões, na forma de explicitar hierarquias e no custo de manutenção, com impactos distintos sobre formulação de consultas, otimização e legibilidade do catálogo.

A estrutura de tabela única corresponde a uma relação não normalizada que reúne, em uma mesma tabela, atributos descritivos e medidas numéricas. Seu apelo está em reduzir o número de junções necessárias para responder a perguntas frequentes, encurtando o caminho de acesso aos atributos referenciados em cláusulas de projeção e seleção. Do ponto de vista lógico, essa organização sacrifica formas normais em favor de simplicidade operacional: há maior redundância e potencial de anomalias de atualização, mas em cenários essencialmente analíticos, com baixa ou nenhuma concorrência de escrita, tal custo pode ser aceitável (ELMASRI; NAVATHE, 2005). Em termos de formulação de consultas, a tabela única tende a favorecer instruções de agregação e filtragem diretas, pois os nomes de colunas e medidas residem em um único catálogo, o que reduz a necessidade de raciocínio sobre chaves e relacionamentos durante a geração do SQL.

O esquema estrela é um arranjo dimensional no qual uma tabela fato central armazena medidas e chaves estrangeiras que a conectam a tabelas dimensão. As dimensões, por sua vez, concentram atributos textuais e hierárquicos que descrevem eixos de análise, como tempo, organização e categorias funcionais. A literatura destaca que essa organização se consolidou em ambientes OLAP por combinar simplicidade de navegação com previsibilidade de junções: a maioria das consultas envolve a tabela fato e um subconjunto de dimensões, o que leva a padrões de junção relativamente estáveis e a expressões claras de agregação e restrição (KIMBALL; ROSS, 2013; PONNIAH, 2010). Embora as dimensões mantenham alguma redundância interna para facilitar cortes e *drill-down*, o esquema estrela privilegia legibilidade e desempenho em leitura, com catálogos que tornam as intenções analíticas mais explícitas no nível lógico (CHAUDHURI; DAYAL, 1997).

O esquema floco de neve é uma variação do desenho dimensional em que as dimensões são parcialmente normalizadas, decompondo atributos em múltiplas tabelas ligadas por chaves estrangeiras. Essa normalização reduz redundância e pode tornar explícitas hierarquias rígidas, melhorando a manutenção quando há reuso

sistemático de níveis descritivos; por outro lado, aumenta o número de junções necessárias e a complexidade das consultas, o que tende a afastá-lo de cenários em que simplicidade e previsibilidade de caminhos de acesso são prioridades (KIMBALL; ROSS, 2013; GOLFARELLI; RIZZI, 2009).

A comparação conceitual entre os três cenários pode ser vista como um compromisso entre simplicidade de consulta, normalização e manutenção. A tabela única minimiza junções e facilita instruções diretas às custas de redundância; o esquema estrela busca um ponto de equilíbrio ao centralizar medidas e oferecer dimensões de leitura clara; o floco de neve privilegia normalização e explicitação de hierarquias, ampliando o número de junções e a complexidade das expressões SQL (CHAUDHURI; DAYAL, 1997; KIMBALL; ROSS, 2013; GOLFARELLI; RIZZI, 2009). Em todos os casos, o catálogo e a nomenclatura dos atributos influenciam o mapeamento semântico entre linguagem natural e elementos do banco, aspecto relevante quando se discutem sistemas que geram consultas a partir de texto.

No plano da fundamentação teórica, esses arranjos oferecem referenciais para avaliar como a estrutura do esquema condiciona o esforço de formulação das consultas e o desenho dos planos de execução. A literatura em armazéns de dados e análise multidimensional sugere que decisões de modelagem devem considerar simultaneamente requisitos de consulta, estabilidade de hierarquias, frequência de mudanças e custos de manutenção, além do comportamento do otimizador do SGBD em face de junções e agregações recorrentes (PONNIAH, 2010; GOLFARELLI; RIZZI, 2009). Esses parâmetros conceituais definem o pano de fundo para discussões metodológicas subsequentes, nas quais diferentes estruturas de catálogo são utilizadas como referência para investigar como variações de complexidade relacional impactam a formulação e a execução de consultas.

2.4 LLMS PARA *TEXT-TO-SQL*

Os grandes modelos de linguagem representam um dos avanços mais significativos da inteligência artificial nas últimas décadas. Baseados em arquiteturas de redes neurais profundas, especialmente os *Transformers*, esses modelos são treinados em grandes volumes de dados textuais com o objetivo de compreender e gerar linguagem natural de forma contextualizada. Sua capacidade de generalização permite a realização de tarefas complexas, como tradução automática, resumo de

textos, respostas a perguntas e, de forma mais recente, a geração de consultas SQL a partir de linguagem natural.

O presente trabalho fundamenta-se nessa evolução tecnológica, explorando a aplicação e o comportamento de cinco modelos de linguagem contemporâneos — Gemma (Google)[2], OpenChat (OpenChat/OpenLLM)[3], Gaia (Ceia/UFG)[4], DeepSeek (DeepSeek AI)[5] e Qwen (Alibaba)[6] — na tarefa de transformação de texto em SQL (*Text-To-SQL*). A seguir, são apresentados os aspectos conceituais, históricos e técnicos que sustentam o uso desses modelos.

2.4.1 Histórico

O Processamento de Linguagem Natural (PLN), subcampo da inteligência artificial, tem por objetivo permitir que computadores compreendam, interpretem e gerem linguagem humana. De acordo com Stryker e Holdsworth (2024), essa área busca aproximar as demandas linguísticas humanas das capacidades computacionais das máquinas, promovendo uma interação mais fluida e acessível entre ambos.

O avanço expressivo do PLN ocorreu a partir de 2017, com a publicação do artigo *Attention is All You Need* (VASWANI *et al.*, 2017), que introduziu o mecanismo de atenção e a arquitetura *Transformer*. Essa inovação permitiu que modelos de linguagem processassem contextos longos de forma paralela, superando as limitações das redes recorrentes (*RNNs* e *LSTMs*), que tratavam as palavras em sequência. A partir desse marco, a área experimentou um crescimento exponencial, com a emergência de modelos cada vez maiores e mais precisos.

Entre os principais modelos desenvolvidos a partir dessa tecnologia estão:

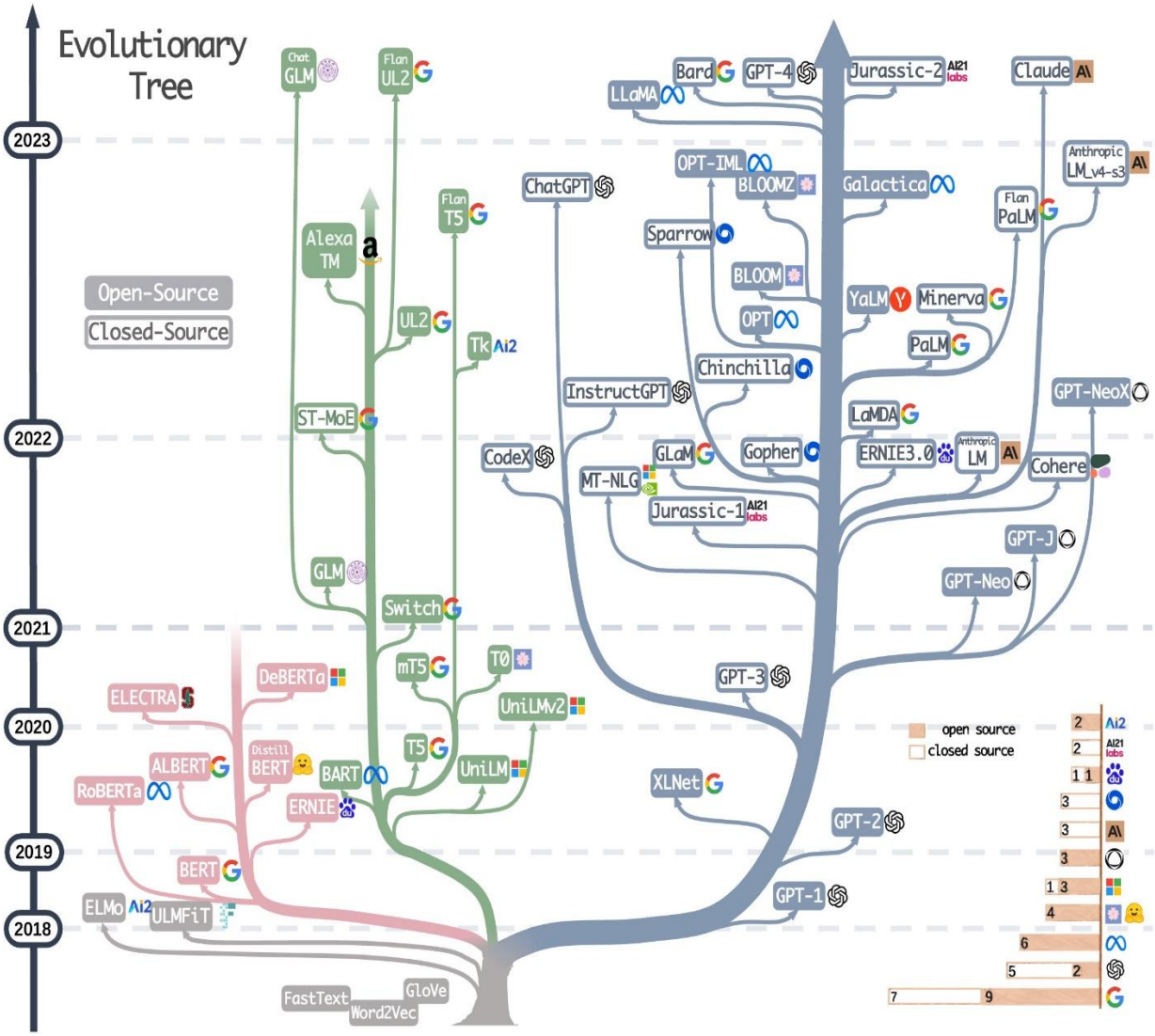
- GPT (RADFORD *et al.*, 2018), que apresentou o uso de pré-treinamento não supervisionado;
- BERT (DEVLIN *et al.*, 2018), que introduziu a aprendizagem bidirecional de contexto;
- XLNet (YANG *et al.*, 2019), que integrou abordagens autoregressivas;

[2] Disponível em: https://huggingface.co/google
[3] https://huggingface.co/openchat
[4] https://huggingface.co/CEIA-UFG
[5] https://huggingface.co/deepseek-ai
[6] https://huggingface.co/Qwen

- T5 (RAFFEL *et al.*, 2020), que unificou múltiplas tarefas de NLP sob a estrutura “texto para texto”;
- e RoBERTa (LIU *et al.*, 2019), que aprimorou o BERT por meio de técnicas otimizadas de treinamento.

Posteriormente, o GPT-3 (FLORIDI; CHIRIATTI, 2020; SCAO *et al.*, 2022) consolidou a era dos grandes modelos de linguagem, destacando-se pela escala massiva e pela capacidade de generalização em múltiplas tarefas sem ajustes específicos. O surgimento de modelos de código aberto e multilíngues ampliou ainda mais o alcance dessa tecnologia, favorecendo pesquisas aplicadas e comparativas, como as conduzidas neste trabalho. A Figura 1 ilustra o processo de evolução dos principais modelos e suas ramificações ao longo dos anos.

Figura 1 - Arvore de evolução dos grandes modelos de Linguagem

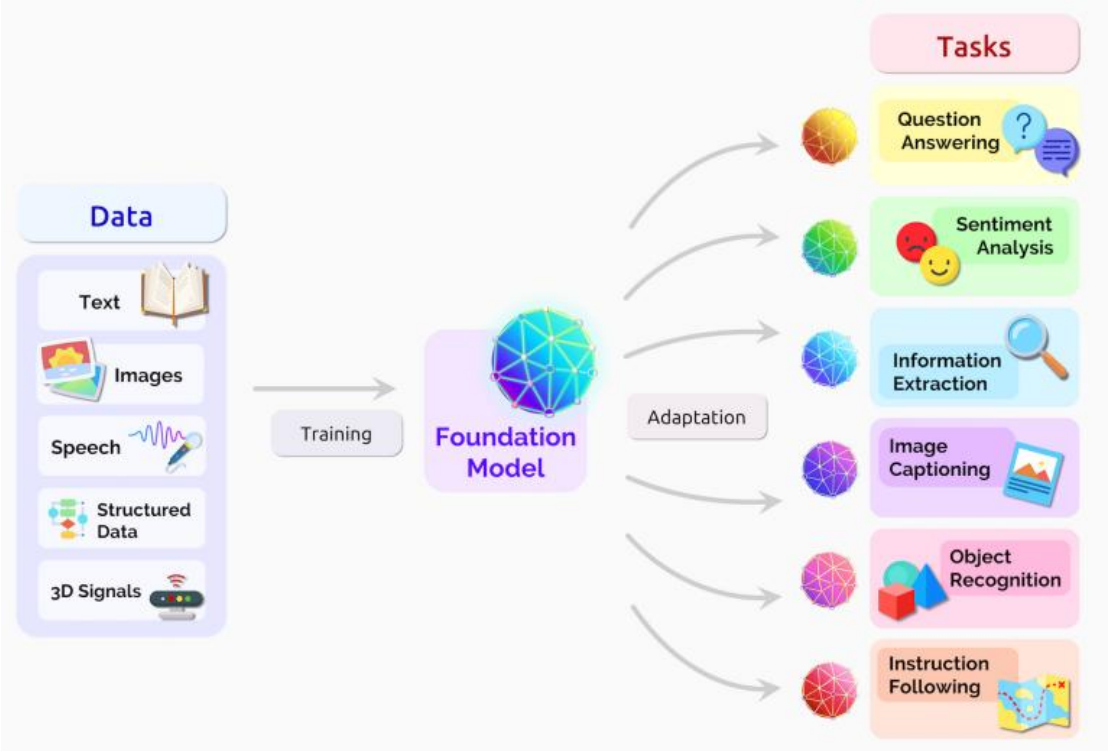


Fonte: https://github.com/Mooler0410/LLMsPracticalGuide/blob/main/imgs/models-colorgrey.jpg

2.4.2 Aplicações

Segundo Bommasani *et al.* (2023), os grandes modelos de linguagem (LLMs) evoluíram significativamente, expandindo suas capacidades de compreensão textual para integrar fontes de informação multimodais, como imagens, áudios e vídeos. Essa transição para dados multimodais não apenas amplia o escopo dos modelos, mas também proporciona uma integração mais rica de contextos, possibilitando a realização de tarefas complexas. A diversidade de entradas, portanto, permite a criação de uma variedade de saídas e a aplicação desses modelos em uma gama de contextos práticos. Entre as aplicações mais notáveis, estão o reconhecimento e a geração de imagens, análise de sentimentos em textos, tradução automática e sistemas de recomendação, entre outros. O conceito de "*Foundation Models*", ou modelos de base, surge dessa evolução, em que os modelos não são projetados apenas para uma tarefa específica, mas têm a capacidade de serem aplicadas técnicas de *fine-tunning* para múltiplas tarefas com alta eficácia. A Figura 2 ilustra essa arquitetura, destacando a estrutura de dados e as aplicações que se beneficiam dessa abordagem flexível.

Figura 2 - Estrutura de dados e aplicações dos modelos de base



Fonte: BOMMASANI *et al.* (2023).

Com a possibilidade de geração de grandes volumes de dados e da adaptação de grandes modelos de linguagem a diferentes contextos, essas tecnologias puderam ser aplicadas de forma eficaz em diversas áreas do conhecimento e do mercado. Na medicina, por exemplo, os modelos de linguagem têm sido utilizados para acelerar diagnósticos, interpretar exames médicos e até mesmo para a descoberta de medicamentos (Thirunavukarasu *et al.*, 2023). No campo da tradução de textos, os modelos oferecem soluções mais rápidas e precisas, melhorando a comunicação em diferentes idiomas (Brants *et al.*, 2023). Na educação, esses modelos estão sendo aplicados para personalizar o ensino, identificar lacunas no aprendizado e até mesmo para criar tutores virtuais interativos, que adaptam o conteúdo conforme a necessidade do estudante (Kasneci *et al.*, 2023). Em tarefas de análise de sentimentos, esses modelos se destacam pela precisão em capturar nuances de sentimentos em textos, algo essencial para empresas que buscam entender melhor o feedback de consumidores (Yang *et al.*, 2024). Na área do direito, os LLMs ajudam a acelerar a revisão de documentos legais e na classificação de jurisprudência, tornando processos complexos mais ágeis e eficientes (Lai *et al.*, 2024). Além disso, na automação, esses modelos têm sido aplicados para otimizar

processos industriais e operacionais, além de permitir a criação de assistentes virtuais avançados (Su *et al.*, 2024). Na área de transformação de texto em SQL, no qual esse trabalho está inserido, esses modelos têm facilitado a criação automática de consultas a partir de instruções em linguagem natural, simplificando a interação com bancos de dados (Shi *et al.*, 2024). O avanço contínuo dessas tecnologias promete expandir ainda mais seu impacto, oferecendo soluções inovadoras para diversos desafios globais.

2.4.3 Grandes Modelos de Base

Os modelos considerados nesta dissertação pertencem a famílias abertas e amplamente documentadas na literatura técnica recente, disponibilizadas em repositórios públicos como o *Hugging Face* e usualmente acompanhadas de variantes instrucionais e suporte multilíngue. Em termos conceituais, tais famílias ilustram o estado da arte em modelos baseados em *Transformer*, combinando pré-treinamento em corpora extensos com etapas de alinhamento por instruções, o que favorece a produção de saídas estruturadas e aderentes a formatos específicos de tarefa, como a geração de consultas SQL.

A linha Gemma (GEMMA TEAM, 2024), mantida por iniciativa corporativa, representa a tendência de disponibilização de modelos abertos orientados à eficiência de inferência e ao uso responsável. Essas distribuições costumam oferecer variantes com diferentes vocações linguísticas, incluindo configurações alinhadas ao português, e são frequentemente empregadas em estudos que investigam a relação entre adaptação instrucional e conformidade de saída em tarefas de geração estruturada.

A família OpenChat (WANG *et al.*, 2023), associada ao ecossistema de modelos alinhados por instruções a partir de bases amplamente difundidas, aparece de forma recorrente em relatórios técnicos que descrevem *pipelines* de curadoria de instruções e refinamento de diálogo. Do ponto de vista conceitual, trata-se de uma linha representativa para discutir como o alinhamento supervisionado e técnicas correlatas influenciam a aderência do modelo a formatos de resposta esperados.

O projeto Gaia (CEIA-UFG, 2025), desenvolvido por grupo acadêmico nacional, oferece uma adaptação instrucional com foco explícito em português do Brasil. A documentação pública enfatiza a curadoria de dados e a orientação das saídas para formatos estruturados, o que torna essa família especialmente pertinente quando se

discute a influência da proximidade linguística entre etapas de ajuste instrucional e domínios de aplicação em língua portuguesa.

A série Qwen (YANG *et al.*, 2024), mantida por um laboratório industrial, reúne modelos multilíngues de propósito geral publicados em iterações sucessivas e frequentemente acompanhados de variantes instrucionais. Esses modelos são citados na literatura técnica como exemplos de combinações entre vocabulários amplos, cobertura de múltiplas línguas e mecanismos de alinhamento que favorecem tarefas de tradução semântica entre linguagem natural e representações formais.

A linha DeepSeek (GUO *et al.*, 2025), por sua vez, inclui modelos obtidos por destilação e de treinamento que buscam explicitar etapas internas de raciocínio mantendo custo de inferência reduzido, aplicando técnica na qual um modelo “aluno”, menor e mais barato, é treinado para imitar o comportamento de um modelo “professor”. No plano teórico, essa família é útil para discutir o papel da destilação e do alinhamento na consistência de saídas estruturadas, bem como os compromissos entre capacidade de representação e eficiência.

Em conjunto, essas famílias permitem fundamentar três aspectos que são centrais para a tarefa de *Text-To-SQL* em contextos bilíngues: a utilidade de variantes instrucionais para aumentar a conformidade com o formato de saída; a importância do suporte multilíngue para a transferência entre perguntas e esquemas em línguas distintas; e o papel de abordagens de eficiência, como destilação e quantização relatadas na documentação pública, para viabilizar o uso de modelos abertos em ambientes computacionais restritos. Essa caracterização, em nível conceitual, estabelece o pano de fundo necessário para as discussões metodológicas e analíticas desenvolvidas nos capítulos subsequentes.

Além das famílias já apresentadas, a literatura descreve outras linhas amplamente empregadas em pesquisas com modelos de pesos abertos e instrução. A série LLaMA consolidou o uso de pesos abertos como base para derivações e ajustes instrucionais (TOUVRON *et al.*, 2023); a família Mistral, incluindo variações de mistura de especialistas, enfatiza eficiência e qualidade em portes reduzidos (JIANG *et al.*, 2023); o projeto BLOOM destaca-se pelo caráter verdadeiramente multilíngue e pela abertura de dados e pesos em larga escala (SCAO *et al.*, 2022), entre outras não citadas aqui, que por questões como tempo e capacidade de análise, estão limitadas.

2.4.4 Hugging Face

A *Hugging Face*[7] constitui uma infraestrutura para pesquisa e desenvolvimento em aprendizado de máquina que integra repositórios públicos de modelos e dados, bibliotecas de uso amplo e serviços de execução em um único ecossistema coerente, promovendo transparência, reprodutibilidade e colaboração científica. Seu repositório de modelos com versionamento e cartões padronizados de documentação, somado a recursos análogos para conjuntos de dados, viabiliza rastreabilidade de experimentos e comparabilidade entre abordagens. No plano técnico, a biblioteca *Transformers* oferece interfaces unificadas para modelos contemporâneos e se articula com componentes como *Datasets*, *Tokenizers*, *Accelerate* e métodos de ajuste fino eficiente, possibilitando desde experimentação exploratória até *pipelines* de produção com custos computacionais controlados. Em paralelo, ambientes de demonstração e serviços de inferência permitem validar hipóteses e disseminar resultados, preservando licenças e metadados. Para esta dissertação, esse conjunto de ferramentas e práticas fornece uma base padronizada para seleção, adaptação e avaliação de modelos multilíngues aplicados à transformação de linguagem natural em SQL, com documentação dos artefatos utilizados e condições de replicação por terceiros.

2.4.5 Estratégias de Aplicação e Inferência

Os LLMs podem ser aplicados em diferentes modos de uso, dependendo da quantidade de exemplos e da adaptação ao domínio da tarefa:

- *Zero-shot learning* caracteriza-se pela execução de uma tarefa sem a apresentação de exemplos específicos, apoiando-se no conhecimento adquirido no pré-treinamento; essa capacidade de transferência foi documentada em linguagem natural por Radford *et al.*, ao demonstrar desempenho *zero-shot* em múltiplas tarefas, e consolidada por Brown *et al.*, que avaliam LLMs em modos *zero-shot* e de aprendizado em contexto. (RADFORD *et al.*, 2019; BROWN *et al.*, 2020).

[7] Disponível em: https://huggingface.co/

- *Few-shot learning* (BROWN *et al.*, 2020): o modelo recebe poucos exemplos de entrada e saída como guia antes de gerar suas próprias respostas, ajustando-se minimamente ao contexto.
- *Fine-tuning* (HOWARD; RUDER, 2018; DEVLIN *et al.*, 2018): envolve o reprocessamento do modelo em um novo conjunto de dados supervisionado, de modo a especializá-lo em uma tarefa ou domínio específico.

No presente estudo, todos os experimentos foram conduzidos em modo *zero-shot*, com o objetivo de avaliar a capacidade de generalização intrínseca dos modelos diante da tarefa de transformar linguagem natural em SQL, sem qualquer tipo de ajuste supervisionado prévio. Essa escolha possibilita observar o desempenho nativo de cada modelo em cenários multilíngues e com diferentes estruturas de banco de dados.

2.4.6 *Conversão de texto para linguagem SQL*

O *Text-To-SQL* é uma área de pesquisa que busca converter perguntas em linguagem natural em consultas SQL válidas. Essa transformação visa eliminar barreiras técnicas para o acesso a dados, permitindo que usuários sem conhecimento em SQL realizem consultas de forma intuitiva. O processo envolve etapas de interpretação semântica e geração estruturada, nas quais o modelo precisa compreender a intenção da pergunta e mapear corretamente os elementos da linguagem natural (entidades, atributos, condições) para o esquema do banco de dados.

De acordo com Shi *et al.* (2024), os avanços recentes em LLMs transformaram o *Text-To-SQL* em um campo de experimentação ideal para avaliar a capacidade de raciocínio simbólico e generalização de modelos de linguagem. Com o uso de arquiteturas baseadas em atenção, os modelos atuais conseguem lidar com múltiplas tabelas, junções complexas e consultas agregadas, superando limitações das abordagens anteriores baseadas em regras ou aprendizado simbólico. A evolução dessa área reflete a convergência entre aprendizado profundo e linguagens estruturadas, tornando o *Text-To-SQL* uma aplicação prática e relevante da inteligência artificial no contexto de acesso a dados públicos e corporativos.

A área de *Text-To-SQL* emergiu com a necessidade de facilitar o acesso a dados em sistemas de gerenciamento de banco de dados, especialmente em

contextos em que usuários não técnicos precisavam interagir com esses sistemas. Nos primeiros estudos, os sistemas eram baseados em regras ou abordagens limitadas, mas com o avanço da inteligência artificial, especialmente com o surgimento de modelos de linguagem de grande escala, o *Text-To-SQL* passou a contar com abordagens mais sofisticadas, baseadas em aprendizado de máquina e redes neurais, aumentando significativamente a precisão e a flexibilidade.

A interface LUNAR [Woods *et al.*, 1972] foi uma das pioneiras no desenvolvimento de sistemas de acesso a bancos de dados. Criada para uso na NASA, sua principal função era permitir consultas a um banco de dados contendo informações sobre amostras lunares coletadas pelo programa espacial Apollo. Já na década de 1980, surgiu o CHAT-80 [Warren & Pereira, 1980], o CHAT-80 utilizava a linguagem PROLOG em todas as etapas do processamento, que incluíam análise sintática, análise semântica e determinação do escopo de quantificadores, demonstrando a aplicação da lógica computacional em consultas a bancos de dados.

Em 1994, o ATIS *(Air Travel Information System)* de Dahl *et al.* (1994), surgiu com um novo trabalho relacionado a transformação de texto em *SQL,* trazia consigo um novo *corpus* preparado de consultas *SQL* e que permitiu o *benchmark* geral para diversos modelos de NLP. Neste *corpus* existiam tantos as consultas em *SQL* como também os textos que dariam origem aos comandos. No entanto, estava limitado a um único domínio, o de viagens áreas. Esse projeto estabeleceu uma base sólida para pesquisas futuras em sistemas de *Text-To-SQL* e NLP, influenciando benchmarks modernos como WikiSQL e Spider.

2.4.7 Bases de dados para *Text-To-SQL*

Entre as bases de dados mais relevantes para o desenvolvimento e avaliação de modelos de *Text-To-SQL*, destacam-se:

- ATIS (Dahl *et al.*, 1994): *corpus* voltado a consultas no domínio de viagens aéreas, pioneiro na padronização de dados para o treinamento de sistemas de linguagem natural.
- WikiSQL (Zhong; Xiong; Socher, 2017): conjunto de pares texto–SQL extraídos da *Wikipedia*, voltado a consultas simples de tabela única.
- Spider (Yu *et al.*, 2018): *benchmark* que abrange múltiplos domínios e consultas complexas, incluindo junções e funções agregadas.

- Bird (Li *et al.*, 2024): base mais recente, que introduz elementos de linguagem natural não estruturada e contexto híbrido para consultas realistas.

A base Spider foi utilizada como referência neste trabalho por ser amplamente adotada pela comunidade científica e por conter consultas de múltiplas tabelas e domínios, além de pares textuais anotados manualmente, o que garante qualidade e diversidade. A partir foi projetada a construção da base de questões e respostas relatadas na seção 4.1.

2.4.7.1 Spider

O *Spider* de Yu *et al.* (2018) é a base escolhida para construção dos testes e avaliação neste trabalho ele é um grande conjunto de dados rotulado por humanos para análise semântica complexa e entre domínios e tarefas de texto para SQL (interfaces de linguagem natural para bancos de dados relacionais). Ele foi lançado junto com o artigo EMNLP 2018: *Spider: A Large-Scale Human-Labeled Dataset for Complex and CrossDomain Semantic Parsing and Text-To-SQL*[8] e será usado como base nas seções seguintes serão desenvolvidas.

A base de dados Spider, apresentada por Yu *et al.* (2018), é amplamente reconhecida como uma das mais importantes no domínio de tarefas de *Text-To-SQL*. Foi projetada com o objetivo de oferecer um conjunto diversificado e complexo de dados para avaliação de modelos de parsing semântico e tradução de texto em consultas SQL. Essa base é única por sua abordagem centrada em múltiplos domínios e pela inclusão de consultas SQL que variam em complexidade, desafiando os modelos a generalizarem suas capacidades de compreensão e geração de linguagem.

O diferencial do Spider reside em características objetivas que o tornam adequado para a avaliação de modelos de linguagem em tarefas de Text-To-SQL. Em termos de escala, a base reúne mais de 10.000 pares de exemplos, nos quais consultas SQL são associadas a descrições em linguagem natural, distribuídos em mais de 200 esquemas de banco de dados pertencentes a diferentes domínios. Essa

[8] O conjunto de dados Spider está disponível no site Yale LILY Lab: https://yale-lily.github.io/spider

abrangência estrutural e temática reduz o risco de viés associado a domínios específicos e permite avaliar a capacidade dos modelos de generalizar para esquemas e contextos não vistos durante o treinamento.

No que se refere à qualidade e à precisão, o Spider é integralmente rotulado por especialistas humanos, o que assegura a correspondência semântica entre as perguntas em linguagem natural e as consultas SQL associadas. Esse processo de anotação manual minimiza ambiguidades e inconsistências, fornecendo um conjunto de referência confiável para a avaliação objetiva da acurácia dos modelos. Além disso, a base foi concebida para incluir consultas que exigem operações complexas, como junções entre múltiplas tabelas, agregações e subconsultas, o que a diferencia de bases como o *WikiSQL*, predominantemente composto por consultas simples sobre tabelas únicas.

Por fim, a relevância do Spider decorre de seu alinhamento com desafios encontrados em aplicações reais de Text-To-SQL. Conforme destacado por Yu et al. (2018), a base foi projetada para avaliar a capacidade dos modelos de compreender esquemas de dados previamente não vistos e de gerar consultas que demandam raciocínio lógico sobre múltiplas relações. Essas características tornam o Spider particularmente adequado para estudos que investigam a robustez e a generalização de modelos de linguagem em cenários realistas.

2.4.8 Métricas de Avaliação

A avaliação de desempenho em tarefas de *Text-To-SQL* é tradicionalmente realizada por meio de métricas apresentadas por Yu *et al.* (2018), que comparam a consulta gerada pelo modelo com a consulta de referência. As mais utilizadas são:

- *Execution Accuracy* (EA): mede se a consulta gerada produz o mesmo resultado que a consulta correta ao ser executada no banco de dados;
- *Logical Form Accuracy* (LF): avalia a equivalência estrutural e semântica entre a consulta gerada e a consulta alvo;
- *Syntactic Validity*: verifica se a consulta é sintaticamente válida e executável.

Essas métricas fornecem uma visão abrangente sobre o desempenho dos modelos, permitindo identificar tanto erros de compreensão semântica quanto limitações na construção sintática do SQL, sendo estas utilizadas nesta pesquisa para

avaliação dos resultados apresentados no capítulo 5, unificadas em uma única métrica.

2.4.9 Princípios de *prompting* para geração estruturada

A formulação de *prompts* exerce influência direta sobre a qualidade e a conformidade sintática de saídas estruturadas como SQL. Três princípios gerais emergem da literatura. O primeiro é a instrução explícita e restritiva, que especifica de forma clara o objetivo, o formato esperado e eventuais limitações, reduzindo a entropia do espaço de saída; tal orientação encontra respaldo na evidência de aprendizado em contexto, pela qual grandes modelos ajustam seu comportamento a partir de exemplos e instruções no próprio prompt (BROWN *et al.*, 2020).

O segundo princípio é a decomposição do raciocínio em etapas latentes, isto é, a externalização do processo de resolução em passos que organizem a seleção de colunas, a determinação do caminho de junção e a aplicação de filtros e agregações; a literatura mostra que encorajar cadeias de raciocínio melhora a consistência e a exatidão em tarefas que exigem múltiplas operações simbólicas (WEI *et al.*, 2022), efeito que pode ser reforçado por estratégias de auto consistência, agregando amostras de raciocínio para estabilizar a resposta final (WANG *et al.*, 2022).

O terceiro princípio é a ancoragem no esquema, que consiste em serializar de modo compacto e inequívoco as tabelas, colunas e chaves relevantes, preferencialmente com nomes canônicos e exemplos mínimos de uso; em *Text-To-SQL*, tal serialização atua como contexto delimitado que facilita o *schema linking* e reduz alucinações sobre objetos inexistentes no catálogo (YU *et al.*, 2018). Em ambientes nos quais a validade sintática é crítica, técnicas complementares de decodificação ou verificação podem ser adotadas, restringindo a geração à gramática de SQL ou verificando incrementalmente a conformidade da estrutura produzida, abordagem que demonstrou reduzir substancialmente erros sintáticos em *parsing* semântico (SCHOLAK; SCHUCHER; BAHDANAU, 2021).

Em conjunto, instruções claras, decomposição explícita do raciocínio, serialização do esquema e, quando possível, decodificação ou verificação constrangida constituem um quadro de *prompting* alinhado à produção de consultas corretas e executáveis, sem prejuízo da generalização inerente ao regime *zero-shot*

ou *few-shot* (BROWN *et al.*, 2020; WEI *et al.*, 2022; WANG *et al.*, 2022; SCHOLAK; SCHUCHER; BAHDANAU, 2021; YU *et al.*, 2018).

3 TRABALHOS RELACIONADOS

A busca por trabalhos relacionados ao objetivo do trabalho foi realizada através de um Mapeamento Sistemático da Literatura (MSL) para buscar o estado da arte para a área de pesquisa. Com os resultados do MSL foi possível identificar os trabalhos que mais se aproximam do proposto e assim foi possível conhecer quais técnicas foram utilizadas para servir de guia para a execução dos testes.

3.1 MAPEAMENTO SISTEMÁTICO DA LITERATURA

Inspirados nas pesquisas médicas, Petersen *et al.* (2008) explicam que os mapeamentos sistemáticos são trabalhos secundários que estudam um conjunto de trabalhos primários em uma área de pesquisa, porém de pontos de vista diferentes. Ambos objetivam levantar informações do estado da arte de uma área de pesquisa, sendo fonte de embasamento para elaboração de novas pesquisas na área em questão.

O principal objetivo de um estudo de mapeamento sistemático é fornecer uma visão geral de uma área de pesquisa, e identificar a quantidade e o tipo de pesquisa e resultados disponíveis dentro dela. Frequentemente, alguém quer mapear as frequências de publicação ao longo do tempo para ver tendências. Um objetivo secundário pode ser identificar os fóruns nos quais a pesquisa na área foi publicada. (PETERSEN *et al.*, 2008).

As etapas essenciais do processo do estudo de mapeamento sistemático são a definição de questões de pesquisa, conduzir a busca por artigos relevantes, triagem de artigos, palavras-chave de resumos e extração e mapeamento de dados. Cada etapa do processo tem um resultado, o resultado do processo sendo o mapa sistemático. (PETERSEN *et al.*, 2008).

3.2 PERGUNTAS NORTEADORAS

As perguntas devem refletir o objetivo do mapeamento sistemático (PETERSEN *et al.*, 2008), desta forma para esse estudo temos as seguintes perguntas:

a) Existe quantidade mínima de 5 publicações nos últimos 5 anos relacionadas à transformação de linguagem natural em consultas SQL? Justificativa: Esta questão visa entender a amplitude e a evolução do tema na literatura acadêmica. Publicações recentes são indicativas de um campo ativo e em desenvolvimento.

b) Entre os trabalhos, existem modelos de linguagem que permitem a escrita em língua portuguesa? Justificativa: A identificação de modelos que operam na língua portuguesa permite que isso garante a aplicabilidade e a acessibilidade das tecnologias para usuários de língua portuguesa.

c) Quais são as bases de dados mais comumente utilizadas para treinamentos de modelos ou customização de modelos de linguagem para transformação de texto em consultas sql? Justificativa: Conhecer as bases de dados utilizadas permite uma compreensão mais aprofundada sobre as metodologias aplicadas e a validade dos resultados obtidos pelos modelos.

d) Quais são as técnicas de linguagem natural que são mais frequentemente utilizados em tarefas de transformação de texto em consultas sql? Justificativa: Avaliar o os modelos existentes ajudam a destacar aqueles que se mostram mais utilizados na tarefa de conversão de linguagem natural para SQL.

3.3 ESTRATÉGIA DE BUSCA

Para a realização da busca bibliográfica, foram selecionadas quatro bases de dados reconhecidas por sua confiabilidade e qualidade, conforme apontado por Buchinger, Cavalcanti e Hounsell (2014). As bases escolhidas incluem IEEE Xplore, Periódicos Capes, ACM Digital Library e arXiv, todas acessíveis de forma gratuita por meio da VPN da UDESC. Essa seleção visou garantir a abrangência e a relevância dos resultados, considerando a expertise técnica e científica disponível nesses repositórios.

A construção da estratégia de busca baseou-se em palavras-chave específicas ao tema central da pesquisa, como "*Text-To-SQL*", "*nl2sql*" e "*natural language to sql*". Por ser um campo majoritariamente técnico, onde a maioria dos trabalhos relevantes é redigida em inglês, optou-se por não incluir sinônimos em português. Com essas palavras-chave, foi formada a *string* de busca: *("Text-To-SQL" OR "nl2sql" OR "natural language to sql").*

Essa *string* foi aplicada nas bases selecionadas, com o objetivo de localizar trabalhos diretamente relacionados ao tema.

Além disso, foram definidos critérios objetivos e subjetivos de inclusão e exclusão para assegurar que os artigos recuperados estivessem alinhados com os objetivos da pesquisa. Os critérios objetivos determinaram que os trabalhos deveriam ser da área de ciência da computação, publicados entre janeiro de 2019 e outubro de 2024, disponíveis em inglês ou português, acessíveis gratuitamente e contendo as palavras-chave no título ou resumo. Já os critérios subjetivos orientaram a seleção de artigos que tratassem especificamente do tema *Text-To-SQL* e contribuíssem significativamente para a discussão do tópico.

Por outro lado, os critérios de exclusão ajudaram a eliminar artigos duplicados e aqueles que não fossem fontes primárias. Esses cuidados na elaboração da estratégia de busca garantiram a obtenção de um conjunto de dados relevante, atual e diretamente relacionado à pesquisa, proporcionando uma base sólida para o desenvolvimento do estudo.

3.4 ANÁLISE DOS RESULTADOS

A análise dos MBAs resultou em uma variação no número de publicações recuperadas: o arXiv apresentou 273 publicações, seguido pelo ACM com 42, Periódicos Capes com 38 e o IEEE com 6 publicações. Ao todo, foram obtidos 359 resultados. Esses números indicam uma concentração maior no arXiv que se destaca como a principal fonte. Foi aplicado o método de busca inicialmente pelo título e resumo dos artigos e após aplicados os critérios de exclusão.

Um ponto importante, são os dados extraídos dos artigos, sendo assim a análise estava em busca de 3 eixos, se se tratava de um artigo que implementa um modelo de linguagem natural voltado para *Text-To-SQL*, se relatava uma base de dados para utilização ou se relatava a aplicação do idioma português. E a seguir podemos observar um resumo dos artigos que mais contribuíram para responder as perguntas norteadoras.

3.4.1 Resumo dos Trabalhos

José *et al.* (2022), explora a integração de sistemas de pergunta e resposta, *Question Answering* – QA, com *Text-to-SQL* para a língua portuguesa. Os autores propõem um modelo que combina técnicas de QA e tradução de linguagem natural para SQL, buscando melhorar a precisão e a fluência das consultas geradas a partir de perguntas em português. A abordagem proposta visa não apenas converter perguntas em consultas SQL, mas também fornecer respostas diretas baseadas nas informações extraídas do banco de dados. Os autores utilizaram um conjunto de dados especificamente desenvolvido para avaliar a eficácia do modelo em situações reais, levando em consideração as nuances e desafios da língua portuguesa. Um dos diferenciais desse trabalho é a ênfase em como as interações entre QA e *Text-To-SQL* podem ser otimizadas para gerar resultados mais relevantes, além de promover uma experiência de usuário mais intuitiva.

Os resultados mostram que o modelo integrado conseguiu melhorar significativamente a precisão das consultas geradas e a relevância das respostas fornecidas, em comparação com abordagens que tratam QA e *Text-To-SQL* como tarefas separadas. A pesquisa destaca a necessidade de avanços na combinação dessas tecnologias para otimizar a compreensão e a interação com bancos de dados em português, contribuindo para a acessibilidade e a utilidade de sistemas de informação. Em relação ao presente trabalho, observa-se uma diferença metodológica central: enquanto José et al. (2022) propõem e avaliam uma arquitetura específica treinada para o português, o projeto aqui desenvolvido concentra-se na avaliação comparativa de modelos de linguagem de grande porte em cenários *zero-shot*, considerando variações simultâneas de idioma e estrutura de banco de dados. Assim, este trabalho complementa a literatura ao investigar até que ponto modelos generalistas conseguem lidar com os mesmos desafios linguísticos, sem treinamento supervisionado específico.

Complementando essa abordagem, Cozman (2021) O artigo apresenta o mRATSQL+GAP, um modelo de transformação de linguagem natural para SQL focado na língua portuguesa. O modelo combina a abordagem mRAT, que utiliza redes neurais para mapear perguntas em linguagem natural a consultas SQL, com a arquitetura GAP. O objetivo é superar desafios comuns na tradução de linguagem

natural para SQL, especialmente considerando as particularidades da língua portuguesa, como a ambiguidade e a variabilidade na expressão das consultas.

Para treinar o modelo, os autores utilizaram um conjunto de dados específico para o português, o que é um diferencial significativo em relação a abordagens anteriores que se concentravam principalmente em inglês. Os resultados obtidos demonstraram que o mRAT-SQL+GAP conseguiu alcançar uma precisão alta nas consultas geradas, superando outros modelos existentes para a língua portuguesa. Essa pesquisa destaca a importância de desenvolver soluções personalizadas para o contexto linguístico local, contribuindo para o avanço das técnicas de *Text-To-SQL* e a inclusão de línguas menos representadas na literatura.

A pesquisa em *Text-To-SQL* também se beneficia de conjuntos de dados robustos, como o Spider, introduzido por Yu *et al.* (2018), um extenso conjunto de dados anotados manualmente, com o objetivo de treinar e avaliar sistemas de *parsing* semântico complexo e *Text-To-SQL*. Composto por 10.181 consultas SQL complexas e 5.693 sentenças em linguagem natural, o Spider abrange 200 esquemas de banco de dados de 138 domínios diferentes, permitindo que os modelos sejam desafiados a generalizar consultas em múltiplos cenários. O diferencial do Spider reside na sua escala e diversidade, superando limitações de conjuntos de dados anteriores que se concentravam em domínios únicos, o que torna a tarefa de *parsing* semântico mais realista e abrangente. Inspirado na filosofia do Spider, o presente trabalho adota múltiplos esquemas e domínios, porém estende esse paradigma ao incorporar variações linguísticas e estruturas de dados alternativas, incluindo esquemas em estrela e tabelas únicas aplicadas a dados públicos brasileiros.

Os resultados mostram que modelos existentes, como Seq2SQL e SQLNet, apresentaram desempenhos baixos ao serem avaliados no Spider, evidenciando a dificuldade das consultas e a necessidade de avanços nas técnicas de *Text-To-SQL*. O *benchmark* Spider destaca-se como uma referência desafiadora, com taxa de precisão insatisfatória para a maioria dos modelos testados, enfatizando a importância de novas abordagens na área.

Li (2023) investiga a capacidade dos LLMs de servirem como interface para bancos de dados, traduzindo linguagem natural em consultas SQL. Para isso, os autores apresentam o BigBench, um *benchmark* de larga escala para *Text-To-SQL* com foco em bancos de dados. O BigBench inclui 214 bancos de dados complexos,

abrangendo diversos domínios e apresentando desafios como tabelas múltiplas, junções e agregações.

O autor avaliou vários LLMs no BigBench, incluindo modelos de código aberto e modelos comerciais. Os resultados mostram que, embora os LLMs tenham apresentado avanços significativos na geração de código SQL, eles ainda enfrentam dificuldades em consultas complexas que exigem raciocínio *multi-hop* e compreensão profunda do esquema do banco de dados. O artigo conclui que, apesar do potencial dos LLMs como interface para bancos de dados, ainda há um longo caminho a percorrer para que eles atinjam um nível de desempenho satisfatório em cenários reais. O BigBench serve como um recurso valioso para a comunidade de pesquisa, permitindo a avaliação e o desenvolvimento de LLMs mais robustos e eficientes para tarefas de *Text-To-SQL*. Embora compartilhe o interesse na avaliação de LLMs para Text-To-SQL, o presente trabalho diferencia-se ao priorizar a análise do impacto do desenho do schema e da correspondência linguística entre pergunta e banco, em vez de focar exclusivamente na complexidade lógica das consultas.

Pourreza *et al.*, (2024) introduz o modelo CHASE-SQL, que aborda a geração de consultas SQL a partir de linguagem natural por meio de raciocínio de múltiplos caminhos e otimização de seleção de candidatos. O CHASE-SQL é projetado para lidar com a complexidade das consultas SQL, onde múltiplos caminhos de raciocínio podem ser necessários para chegar à resposta correta. O modelo visa melhorar a precisão na geração de SQLs, especialmente em cenários onde a pergunta requer um raciocínio mais complexo e consideração de preferências na seleção de candidatos. Em contraste com essa proposta arquitetural, o presente trabalho não incorpora mecanismos explícitos de raciocínio múltiplo, mas fornece evidências empíricas de que a complexidade estrutural do schema exerce influência comparável ou superior à complexidade da pergunta sobre o desempenho dos modelos. Os autores propuseram uma abordagem que integra técnicas de raciocínio múltiplos caminhos para explorar diversas interpretações de uma pergunta em linguagem natural, permitindo que o modelo considere diferentes possibilidades ao gerar a consulta SQL. Além disso, o modelo é otimizado para selecionar candidatos mais relevantes, minimizando o número de tentativas incorretas antes de chegar à resposta certa. Essa metodologia busca melhorar a eficiência do processo de geração de SQL e aumentar a robustez do modelo em relação a perguntas complexas.

CHESS, apresentado por Talaei *et al.* (2024), propõe uma síntese eficiente de SQL, integrando informações contextuais para melhorar a interpretação das intenções do usuário. Os resultados experimentais demonstram que o CHESS gera consultas SQL de qualidade, superando modelos anteriores que não utilizavam contexto de forma eficaz.

Por fim, Dong *et al.* (2024) apresentam o modelo C3, que utiliza o ChatGPT em um cenário de *zero-shot*. Os autores investigaram a capacidade do ChatGPT de gerar consultas SQL precisas a partir de perguntas em linguagem natural, mesmo sem treinamento específico em conjuntos de dados relacionados a SQL. O objetivo do estudo é avaliar se um modelo de linguagem amplamente treinado pode ser adaptado para realizar a conversão de linguagem natural para SQL sem necessidade de exemplos específicos durante o treinamento.

A abordagem *zero-shot* do C3 permite que o modelo utilize seu conhecimento prévio sobre a estrutura da linguagem e a lógica de consultas SQL para interpretar e responder a perguntas em linguagem natural. Os autores testaram o modelo em uma variedade de conjuntos de dados e casos de uso, analisando seu desempenho em termos de precisão e adequação das consultas geradas.

Os resultados mostraram que o C3 alcançou resultados competitivos, mostrando que o *ChatGPT* pode ser uma ferramenta eficaz para *Text-To-SQL* em situações em que dados de treinamento específicos não estão disponíveis. Além disso, o estudo destaca as vantagens da flexibilidade e da generalização dos modelos de linguagem em tarefas de geração de consultas SQL, abrindo novas possibilidades para a aplicação de modelos de linguagem em cenários práticos. Esses estudos ressaltam a evolução contínua das técnicas de *Text-To-SQL*, destacando a importância de considerar as especificidades linguísticas e contextuais para aprimorar a eficácia dessas abordagens. Alinhado a essa abordagem, o presente trabalho também adota um cenário *zero-shot*, porém amplia a análise ao considerar múltiplas combinações de idioma e estrutura de banco de dados, permitindo identificar condições específicas nas quais o desempenho dos modelos é favorecido ou degradado.

3.4.2 Respostas às perguntas norteadoras

A análise realizada sobre os trabalhos publicados nos últimos cinco anos confirma a existência de, pelo menos, sete publicações relacionadas à transformação de linguagem natural em consultas SQL, no período de 2019 a 2024. Entre essas publicações destacam-se os trabalhos de Marcos M. José (2022), Marcelo Archanjo José e Fábio Gagliardi Cozman (2021), Gao (2023), Pourreza (2024), Talaei (2024), Maamari (2024), Dong (2024) e Garcia (2024). Esses estudos representam uma evolução contínua e demonstram o crescimento do interesse acadêmico pela área, atendendo ao critério de quantidade mínima exigida para responder à questão.

No contexto da escrita em língua portuguesa, foi identificado que pelo menos três trabalhos explicitam o desenvolvimento ou adaptação de modelos para atender a esse idioma. O estudo de José *et al.* (2022) explora a integração de sistemas de Pergunta e Resposta (QA) com *Text-To-SQL* para a língua portuguesa, abrindo possibilidades de consultas em bancos de dados por meio de linguagem natural nesse idioma. Já o trabalho de José e Cozman (2021) apresenta o mRAT-SQL+GAP, um modelo robusto para tradução de linguagem natural para SQL em português

As bases de dados mais comumente utilizadas para treinamento e customização de modelos voltados à transformação de texto em consultas SQL incluem, majoritariamente, o Spider proposta por Yu *et al.* (2018). Esta base é amplamente reconhecida por sua diversidade e abrangência, contendo mais de 10 mil consultas SQL complexas, aproximadamente 10 mil sentenças em linguagem natural e 200 esquemas de banco de dados.

Quanto aos modelos de linguagem natural mais frequentemente empregados em tarefas de transformação de texto em consultas SQL, diversos se destacam na literatura recente. O mRAT-SQL+GAP, desenvolvido por José e Cozman (2021), se sobressai pela alta precisão alcançada na tradução de linguagem natural para SQL em português. Outro modelo relevante é o CHASE-SQL (POURREZA *et al.*, 2024), que se baseia em raciocínio *multipath* e demonstra melhorias significativas em perguntas mais complexas. Já o CHESS (TALAEI *et al.*, 2024) apresenta integração de informações contextuais, superando modelos anteriores na qualidade das consultas geradas. Por fim, o modelo C3 (Dong *et al.*, 2024) explora a abordagem *zero-shot* com o ChatGPT, evidenciando o potencial de modelos amplamente treinados para gerar consultas SQL com precisão mesmo sem ajustes específicos.

Esses modelos representam avanços significativos no campo, cada qual com suas peculiaridades e áreas de aplicação específicas.

Em termos de modelo de base, o mRAT-SQL+GAP é um parser neural supervisionado derivado do RAT-SQL+GAP, cujo núcleo usa um codificador transformer (mBART) treinado/fine-tuned com dados anotados (Spider traduzido), portanto não é *zero-shot* por desenho. Já o CHASE-SQL é um arranjo multiagente que orquestra LLMs em tempo de inferência (divide-and-conquer, chain-of-thought, exemplos sintéticos) e emprega um seletor treinado; não define um único LLM “base”, mas uma arquitetura que pode usar diferentes LLMs geradores e um modelo de seleção fine-tuned. O CHESS também é um framework multiagente baseado em LLM (recuperação de contexto, seleção de esquema, geração e testes de unidade), pensado para escalabilidade e privacidade, novamente sem fixar um único modelo base — ele “pluga” LLMs abertos ou proprietários conforme o cenário. Por fim, o C3 é explicitamente *zero-shot* e tem como modelo de base o ChatGPT (família GPT-3.5/4), apoiando-se apenas em engenharia de prompt e calibradores de saída, sem fine-tuning em dados Text-to-SQL.

Dentre as técnicas e modelos analisados, o C3 é o que mais se alinha ao escopo deste projeto por adotar geração de SQL em regime *zero-shot* com o ChatGPT, valendo-se apenas de instruções padronizadas e sem ajuste fino específico em corpora de Text-to-SQL. Essa opção coincide com nossa estratégia experimental, que controla idioma e modelagem do banco (esquema estrela e tabela única) e exige apenas a emissão de SQL compatível com SQLite. Assim, tomamos o C3 como referência conceitual para o delineamento dos *prompts* e do protocolo de avaliação; em seguida, seguiremos com testes sistemáticos usando outros modelos de base para comparação, mantendo o regime *zero-shot*.

3.5 AMEAÇAS A VALIDADE

Esta pesquisa apresenta algumas limitações em potencial que devem ser consideradas ao interpretar os resultados. Primeiramente, os mecanismos de busca utilizados nos anais dos eventos podem ter influenciado os resultados obtidos. A eficácia e abrangência desses mecanismos de busca podem variar, potencialmente levando à omissão de trabalhos relevantes.

Em segundo lugar, o processo de extração e classificação dos dados envolveu intervenção humana, o que introduz a possibilidade de erro humano e viés. Apesar dos esforços para garantir a precisão e consistência, a subjetividade inerente à interpretação e categorização dos dados pode ter afetado os resultados.

Os trabalhos mapeados mostram que a área evoluiu em modelos e técnicas, mas permanece dependente de *benchmarks* em inglês e raramente avalia, de modo controlado, o impacto do idioma e da modelagem do banco. Estudos como Spider impulsionam a pesquisa, porém não cobrem o português nem comparam sistematicamente esquema estrela e tabela única; trabalhos focados em PT avançam no modelo, mas não oferecem uma base ampla, bilíngue e reprodutível. Essa combinação de lacunas se conecta diretamente ao objetivo deste estudo: criar um artefato que permita testar, sob condições comparáveis, como idioma e modelagem influenciam a geração de SQL por LLMs em regime *zero-shot*. Assim, os achados da literatura motivam a construção de uma base bilíngue em SQLite com variantes em estrela e em tabela única e um conjunto de perguntas com SQL de referência nos dois idiomas, além de *prompts* padronizados para reduzir vieses de engenharia de prompt. Esses elementos respondem às ausências observadas no estado da arte e fornecem um terreno comum para que diferentes modelos possam ser avaliados com clareza quanto a idioma e estrutura do esquema.

4 TEXT-TO-SQL EM *ZERO-SHOT* SOBRE DADOS PÚBLICOS

Este capítulo apresenta e contextualiza o desenvolvimento do estudo, iniciando pela problemática que motiva a investigação. Embora os portais de transparência e catálogos de dados abertos tenham ampliado o acesso à informação pública, a extração de respostas específicas ainda pressupõe domínio de classificações orçamentárias, compreensão de dicionários de dados e capacidade de escrever consultas SQL corretas. Para o público leigo, essa combinação de exigências técnicas funciona como barreira de entrada, o que limita a apropriação social dos dados e restringe seu uso em rotinas de análise, fiscalização e tomada de decisão.

Para enfrentar esse problema, este trabalho propõe e materializa um ambiente de avaliação composto por uma base de testes e um conjunto de *prompts* e questões em linguagem natural destinados a aferir o desempenho de diferentes modelos de linguagem em *Text-To-SQL*. A base utiliza dados orçamentários reais do Estado de Santa Catarina e foi organizada em duas representações complementares, um esquema estrela e uma tabela única, com versões bilíngues de esquemas e rótulos para permitir o controle do fator idioma. O conjunto de questões foi elaborado com SQL de referência e verificação por execução, inspirado na tradição de *benchmarks* como o Spider, de modo a cobrir operações recorrentes em consultas públicas, incluindo seleção, filtragens temporais e por classificações, junções dimensionais e agregações usuais no domínio.

Os experimentos foram conduzidos em regime *zero-shot*, sem ajuste fino, exclusivamente com modelos abertos viáveis de operar em infraestrutura local. A avaliação emprega uma métrica única de acurácia composta, adotada em todo o

projeto, segundo a qual uma saída é considerada correta apenas quando alcança simultaneamente equivalência de resultados em execução e correspondência semântica e estrutural com o SQL de referência. Esse desenho experimental isola os efeitos do idioma e do desenho do banco e permite comparar, de forma direta, como diferentes LLMs respondem às variações de cenário típicas do uso em dados público. Sendo assim, as seções seguintes detalham a construção da base e do conjunto de questões, apresentam o protocolo experimental e a infraestrutura de execução

4.1 CONSTRUÇÃO DA BASE DE DADOS

A construção da base de dados bilíngue representou um dos resultados mais relevantes desta pesquisa pois viabiliza os experimentos de avaliação dos modelos de linguagem. A criação dessa base teve como propósito fornecer um ambiente controlado e representativo de consultas reais em contexto governamental, permitindo a análise de desempenho de diferentes arquiteturas de LLMs sob condições estruturais e linguísticas distintas.

Os dados utilizados foram extraídos do Portal da Transparência do Governo do Estado de Santa Catarina[9], abrangendo registros de despesas orçamentárias realizadas entre janeiro e abril de 2025, apresenta-se uma amostra do arquivo original na Tabela 1, com uma fração do conjunto de dados, apenas algumas tabelas foram transcritas para melhor visualização. Essa escolha se deve ao caráter público e verificável das informações, bem como à sua estrutura que oferece um cenário realista para a tarefa de conversão de linguagem natural em SQL. O conjunto original continha dados sobre órgãos, funções, subfunções, programas, naturezas de despesa, fontes de recurso, valores financeiros, entre outros.

Tabela 1 – Dados do Portal da Transparência com colunas selecionadas

Ano	Nro Mês	Mês	Código Órgão	Órgão	Vl Empenhado	Vl Liquidado	Vl Pago
2025	4	Abril	48000	Secretaria de Estado da Saúde	R$ 0,00	R$ 6.184,90	R$ 1.810,70

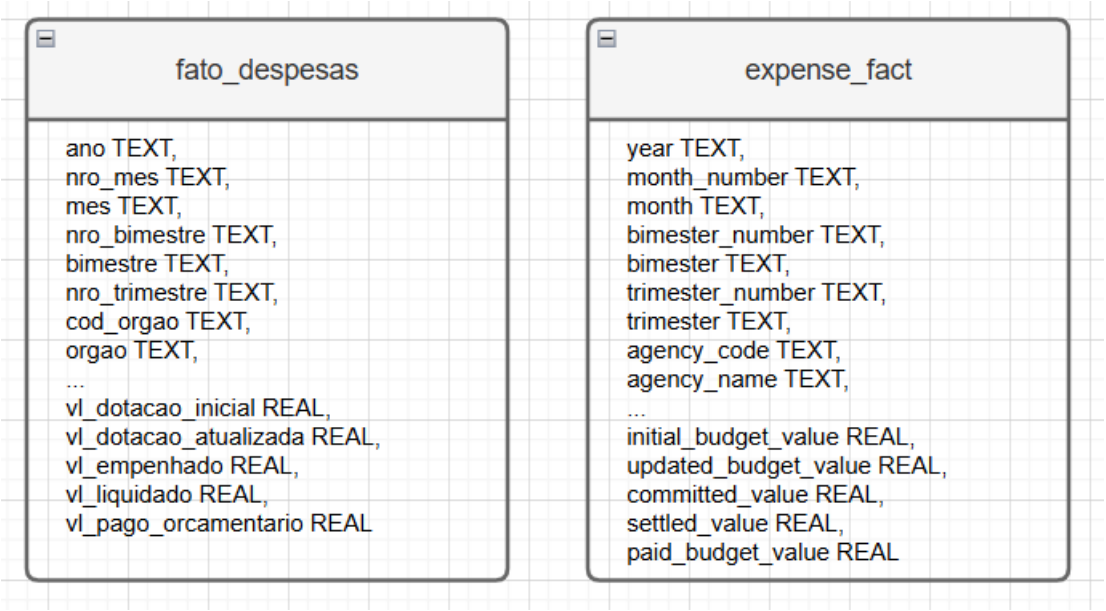
[9] Dados abertos pelo link https://www.transparencia.sc.gov.br/dados-abertos/32/subareainteresse/35

2025	4	Abril	34000	Secretaria de Estado da Comunicação	R$ 0,00	R$ 7.344,25	R$ 7.344,25
2025	4	Abril	15000	Defensoria Pública do Estado de Santa Catarina	R$ 212,00	R$ 0,00	R$ 0,00
2025	4	Abril	29000	Secretaria de Estado de Portos, Aeroportos e Ferrovias	R$ 55.208,00	R$ 55.208,00	R$ 55.208,00
2025	4	Abril	45000	Secretaria de Estado da Educação	R$ 0,00	R$ 8.950,00	R$ 8.950,00

Fonte: elaborado pelo autor

O processo de construção da base foi dividido em quatro etapas principais. A primeira consistiu na coleta e limpeza dos dados, realizada por meio de scripts de extração do arquivo *CSV* disponibilizado pelo portal da transparência.

Figura 3 - Representação das tabelas únicas em português e inglês com campos reduzidos

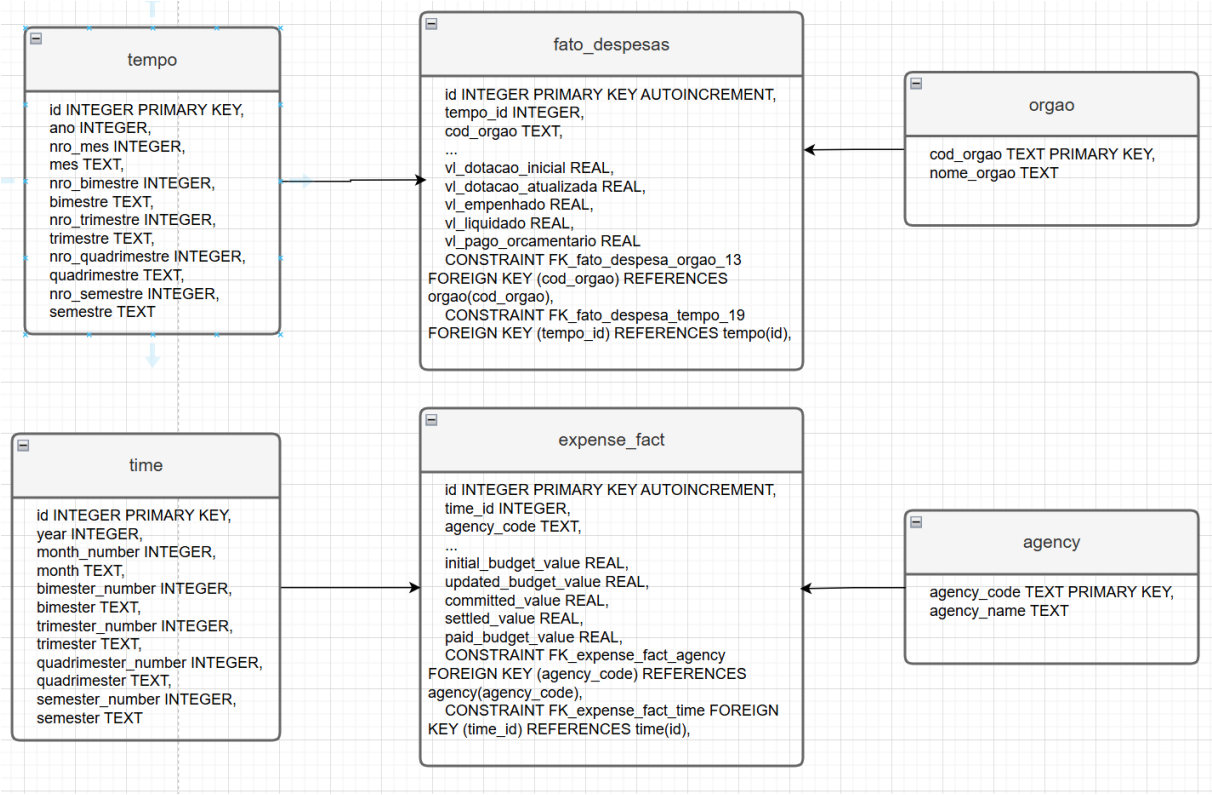


Fonte: Elaborado pelo Autor

A segunda etapa compreendeu a modelagem da estrutura relacional, que deu origem a duas configurações distintas de banco de dados. Na primeira, adotou-se um modelo em esquema estrela, composto por uma tabela fato central — denominada fato_despesa — e múltiplas tabelas dimensionais correspondentes a atributos descritivos, como tempo, orgao, funcao, subfuncao e programa. Essa estrutura foi projetada de modo a refletir a organização típica de bases analíticas empregadas em ambientes de business intelligence e de controle público. Na segunda configuração,

elaborou-se um modelo de tabela única, denominado fato_despesa, no qual todos os atributos foram agregados em uma mesma tabela não normalizada. Essa duplicidade estrutural foi planejada para permitir a comparação entre cenários de diferentes níveis de complexidade, possibilitando verificar o impacto da modelagem sobre a capacidade de interpretação e geração de consultas pelos modelos de linguagem. A Figura 4 ilustra, lado a lado, um recorte da tabela única em português, com medidas orçamentárias e rótulos administrativos, e o núcleo da tabela fato em inglês com medidas homólogas, o que evidencia a equivalência semântica entre as variantes.

Figura 4 - Representação das tabelas no modelo estrela em português e inglês com campos reduzidos



Fonte: Elaborado pelo Autor

A terceira etapa envolveu a tradução integral dos esquemas de dados para o idioma inglês, resultando em um conjunto bilíngue de tabelas e colunas. Nessa fase, os nomes das variáveis foram traduzidos de forma técnica, preservando o significado original e garantindo a correspondência semântica entre as versões. Por exemplo, os campos orgao, funcao e subfuncao foram traduzidos para agency, function e subfunction, respectivamente, enquanto fato_despesa foi convertida em expense_fact. A Figura 5 apresenta uma visão relacional do esquema estrela em

ambos os idiomas, destacando as relações entre a tabela fato e dimensões como tempo e órgão, com chaves primárias e estrangeiras assinaladas. Esse processo permitiu a criação de quatro variações completas de *schema*: (i) estrela em português, (ii) estrela em inglês, (iii) tabela única em português e (iv) tabela única em inglês, as Figuras 4 e 5 materializam esse mapeamento e permitem ao leitor verificar a correspondência terminológica e estrutural utilizada nos testes, porém não representam na totalidade todos os campos e tabelas mapeados, para isso pode se recorrer aos apêndices B, C, D e E de forma descrita e nos apêndices J, K, L e M para visualizar.

A quarta e última etapa consistiu na implementação e validação dos bancos de dados, realizada em ambiente *SQLite*. Foram definidas chaves primárias e estrangeiras, restrições de integridade e índices básicos para garantir o funcionamento relacional adequado das tabelas do modelo estrela. A base final resultante contém aproximadamente 134 mil registros de despesas públicas, estruturados de forma consistente nas quatro versões mencionadas.

Como produto complementar à estrutura física, foi desenvolvido um conjunto bilíngue de cinquenta perguntas em linguagem natural, inspirado em padrões do *benchmark* Spider e adaptado ao contexto das finanças públicas brasileiras. Cada pergunta foi redigida em português e traduzida para o inglês, acompanhada de sua consulta SQL de referência. O conjunto final totaliza duzentos pares de pergunta e SQL, distribuídos entre as quatro combinações possíveis de idioma da questão e idioma do esquema de dados. Para tornar concreta a ideia de equivalência semântica entre variações de idioma e de modelagem, a Tabela 2 apresenta a aplicação de uma mesma pergunta nas quatro configurações. A pergunta busca identificar a ação com maior valor total de dotação inicial no ano de 2025; para garantir comparabilidade entre estruturas, as consultas foram harmonizadas por agregação e ordenação do somatório do valor de dotação, de modo que as diferenças entre versões decorram apenas de nomes de tabelas e colunas e da necessidade de junções no modelo estrela.

Tabela 2 - Aplicação de uma mesma questão nas diferentes estruturas

Domínio	Questão em português	Questão em inglês	Sql Correspondente

Modelo estrela em português	Qual a ação com maior valor de dotação inicial em 2025?	What is the action with the highest initial appropriation in 2025?	**SELECT** a.nome_acao **FROM** fato_despesa f **JOIN** acao a **ON** f.cod_acao = a.cod_acao **JOIN** tempo t on f.tempo_id = t.id **WHERE** t.ano = 2025 **ORDER BY** f.vl_dotacao_inicial **DESC LIMIT** 1;
Domínio	Questão em português	Questão em inglês	Sql Correspondente
Tabela única em português	Qual a ação com maior valor de dotação inicial em 2025?	What is the action with the highest initial appropriation in 2025?	**SELECT** acao, **MAX**(vl_dotacao_inicial) **FROM** despesas_consolidadas **WHERE** ano = '2025';
Modelo estrela em inglês	Qual a ação com maior valor de dotação inicial em 2025?	What is the action with the highest initial appropriation in 2025?	**SELECT** ac.action_name **FROM** expense_fact ef **JOIN** action ac **ON** ef.action_code = ac.action_code **JOIN** time t **ON** ef.time_id = t.id **WHERE** t.year = 2025 **ORDER BY** ef.initial_budget_value **DESC LIMIT** 1;
Tabela única em inglês	Qual a ação com maior valor de dotação inicial em 2025?	What is the action with the highest initial appropriation in 2025?	**SELECT** action_name, **MAX**(initial_budget_value) **FROM** expense_fact **WHERE** year = '2025';

Fonte: Elaborado pelo Autor

A inclusão desta tabela no corpo do capítulo reforça a integração entre texto e figuras anteriormente apresentadas. As Figuras 4 e 5 ilustram, respectivamente, as versões condensadas em tabela única e a organização em esquema estrela, em português e em inglês. A Tabela 2 coloca essas representações em operação por meio de uma pergunta concreta, evidenciando que, no modelo estrela, a obtenção do mesmo resultado exige junções com dimensões como ação e tempo, enquanto, na tabela única, a agregação pode ser feita diretamente sobre colunas não normalizadas. Essa articulação entre visualizações e exemplo executável documenta a equivalência semântica entre variantes e demonstra a adequação do material para avaliar, em cenário *zero-shot*, a sensibilidade dos modelos de linguagem às escolhas de idioma e de modelagem.

O resultado dessa construção constitui uma contribuição para a pesquisa em *Text-To-SQL*, pois disponibiliza um conjunto de dados bilíngue, estruturado e

representativo de um domínio público real. Diferentemente de *benchmarks* tradicionais, como o Spider, que se concentram em bases genéricas e no idioma inglês, a base aqui desenvolvida reflete as especificidades da administração pública brasileira e permite avaliar a performance dos modelos em condições multilíngues e institucionais. Além de servir como instrumento experimental deste trabalho, essa base estabelece fundamentos para estudos futuros de *fine-tuning* e de geração de SQL em língua portuguesa, com potencial de reuso em aplicações voltadas à transparência e ao acesso automatizado a dados governamentais.

Para assegurar transparência e reprodutibilidade, todo o material desta base de questões e respostas — compreendendo a base de dados de despesas públicas, os esquemas em estrela e em tabela única, o conjunto de questões bilíngues, bem como os scripts de preparação — encontram-se disponibilizados em repositório público por meio da plataforma online github[10].

4.2 CONSTRUÇÃO DO *PROMPT*

A formulação do prompt adotado nesta pesquisa buscou maximizar a conformidade estrutural das saídas e reduzir a variabilidade típica de respostas em linguagem natural, de modo a induzir a geração de um único comando SQL válido para o dialeto SQLite, em regime zero-shot e sob diferentes condições linguísticas. Em linha com os princípios discutidos na fundamentação (seção 2.4.9), a estratégia combinou: i) instrução explícita de papel e de formato de saída; ii) delimitação concisa do contexto relevante por meio da serialização do esquema do banco; iii) restrição forte para que o modelo não produza explicações, comentários ou múltiplas consultas; e iv) sentinela de término (ponto e vírgula) para facilitar a validação programática da resposta. Foram empregadas duas variantes semanticamente equivalentes do prompt, uma em português e outra em inglês, a fim de avaliar o comportamento em quatro combinações de idioma entre pergunta e esquema (PT–PT, EN–EN, PT–EN, EN–PT).

No cerne do desenho está a ancoragem explícita no dialeto: declara-se que o sistema é um “gerador de SQL para SQLite”, reduzindo ambiguidade sobre funções e operadores disponíveis. Em seguida, a pergunta do usuário e o esquema do banco

[10] Link de acesso à base de dados https://github.com/Jonasorso/Despesas_Sc_Text-To-Sql

de dados são inseridos como contexto imediato, sem exemplos adicionais (*zero-shot*), e a instrução prescritiva determina que apenas um comando SQL, terminado por ";", deve ser retornado. Acrescenta-se ainda a orientação "não tente calcular valores", cujo objetivo é evitar que o modelo apresente resultados numéricos "resolvidos" (por exemplo, somatórios computados pelo próprio modelo), em vez de produzir a consulta correspondente. Para mitigar ruídos de pontuação, a interrogação final da pergunta é removida e reintroduzida de forma controlada, prevenindo duplicação de sinais que alguns modelos interpretam como marcadores de encerramento de turno.

Os textos abaixo reproduzem os *templates* efetivamente utilizados (variáveis *schema* e *question* são interpoladas no momento da inferência):

Quadro 1 - Exemplo de prompt base em português

```
Português (PT)
"Você é um gerador de SQL para SQLite. "
"Não tente calcular valores, apenas gerar um comando SQL"
"Dada a pergunta e o esquema do banco de dados abaixo, gere **apenas UM comando SQL válido**, sem explicações ou texto extra. O comando deve terminar com ponto e vírgula `;`. Não escreva nada além do SQL.. "
f"Esquema do banco de dados:\n{schema} "
f"Escreva um comando sql valido para a pergunta com base no schema: {question.replace('?', '')}? "
"SQL: ###Resposta: "
```

Fonte: Elaborado pelo Autor

Quadro 2 - Exemplo de prompt base em inglês

```
Inglês (EN)
"You are an SQL generator for SQLite."
"Given the question and the database schema below, generate only **ONE** valid SQL command,with no extra explanations or text. The command must end with a semicolon ';'. Do not write anything other than the SQL."
f"Database schema:\n{schema} "
f"Write an SQL command for the question: {question.replace('?', '')}? "
"SQL: ###Response: "
```

Fonte: Elaborado pelo Autor

Alguns pontos merecem registro por seu impacto na qualidade e na mensurabilidade dos resultados. Primeiro, a presença de rótulos como "SQL: ###Resposta / ###Response" ao final do prompt foi pensada como um marcador semântico do bloco de saída; todavia, certos modelos tendem a ecoar esses rótulos no texto gerado. Para garantir a aderência à métrica de acurácia de execução, implementou-se pós-processamento mínimo que remove eventuais prefixos

alfanuméricos antes do primeiro token SQL reconhecível, valida o ponto e vírgula final e rejeita saídas múltiplas. Segundo a serialização do esquema é deliberadamente compacta e imediata, evitando descrições narrativas; como discutido na fundamentação, isso favorece o *schema linking* ao expor exatamente os nomes de tabelas, colunas e chaves sobre os quais a consulta deve operar. Terceiro, por envolver português com diacríticos e termos administrativos, o prompt não força aspas ou normalização de identificadores; em vez disso, confia na nomenclatura tal como definida no catálogo. Na prática, observou-se que modelos que preservam fielmente a grafia dos identificadores minimizam erros sintáticos; por essa razão, a avaliação exige correspondência literal com o esquema fornecido.

As duas variantes linguísticas são paralelas em estrutura e restrições, diferindo apenas no idioma das instruções. Essa duplicação permite separar o efeito do idioma do enunciado do efeito do idioma do esquema, preservando constante o "molde" pragmático do prompt. Em cenários cruzados (por exemplo, pergunta em português e esquema em inglês), a instrução permanece no idioma do enunciado para reduzir fricção cognitiva do modelo entre instrução e conteúdo; nas combinações homogêneas (PT–PT, EN–EN), todo o contexto permanece monolíngue. Em todos os casos, a referência explícita a SQLite orienta escolhas de funções nativas (por exemplo, date, agregações padrão) e evita derivações de outros dialetos.

Por fim, a construção do prompt foi guiada por dois objetivos metodológicos: reprodutibilidade e mensuração objetiva. A prescrição de "apenas um" comando, a sentinela ";" e a ausência de explicações textuais viabilizam uma checagem binária simples (executa e retorna o gabarito, ou não), ao passo que a inclusão do esquema como único contexto técnico reduz alucinações sobre objetos inexistentes. Essa configuração mantém o regime *zero-shot*, controla a variância de formato entre modelos e estabelece um ponto de comparação estável para analisar diferenças de desempenho decorrentes de idioma e de estrutura de banco (tabela única versus estrela).

Desta forma os Apêndices F, G, H e I registram integralmente os *prompts* utilizados nos experimentos, contemplando as duas modelagens de dados e os dois idiomas adotados. Os Apêndices F e G apresentam, respectivamente em português e em inglês, o prompt para a variante em tabela única, enquanto os Apêndices H e I reúnem os *prompts* para o esquema estrela, também em português e em inglês. Em todos os casos, o enunciado fixa o contexto de geração em SQLite, impõe a produção

de um único comando SQL válido, sem texto adicional, e incorpora o esquema correspondente como parte do próprio prompt, assegurando que a tarefa seja avaliada em condições equivalentes entre idiomas e estruturas.

Este capítulo apresentou a construção da base de dados, as variantes de modelagem e o protocolo de *prompts* adotado para a avaliação. Nas seções seguintes, são expostos os resultados obtidos e desenvolvida a análise crítica desses valores, discutindo implicações, limitações e efeitos do idioma e do esquema na qualidade das consultas geradas.

5 EXPERIMENTOS E RESULTADOS

Para validar a base e os *prompts* construídos, esta seção introduz o conjunto de experimentos realizados e os resultados obtidos na análise do desempenho de modelos de linguagem de grande escala na tarefa de geração de consultas SQL a partir de linguagem natural. A avaliação incide sobre as quatro combinações entre idioma do enunciado e idioma do esquema e sobre as duas estruturas de dados propostas, de modo a estimar o efeito conjunto de escolhas linguísticas e de modelagem na qualidade das saídas. Mantém-se o regime *zero-shot*, com o prompt único descrito na seção anterior 4.2 e a métrica de acurácia adotada como critério de correção, de forma a garantir comparabilidade direta entre cenários.

À luz da fundamentação, o trabalho C3 foi tomado como referência conceitual por formalizar a geração de SQL em regime *zero-shot* com instruções padronizadas. Contudo, por utilizar o ChatGPT — serviço pago — sua adoção direta contrariaria os objetivos de reprodutibilidade e de execução local deste estudo. Assim, optou-se por operacionalizar os mesmos princípios do C3 em um cenário aberto: inferência *zero-shot*, sem *fine-tuning*, com *prompt* único e controle explícito de idioma e de modelagem do banco, mas empregando modelos de abertos executados localmente.

A seleção de Qwen, Gemma, Gaia, OpenChat e DeepSeek decorre, portanto, de uma convergência entre coerência metodológica e viabilidade prática. Mantiveram-se modelos com distribuição oficial no *Hugging Face*, variantes instrucionais e suporte efetivo a português ou a cenários multilíngues, além de porte compatível com a infraestrutura disponível. Todos os experimentos foram conduzidos em um único nó com Intel Core i5-13500, 32 GiB de RAM, SSD NVMe de 512 GB e GPU NVIDIA RTX 5000 Ada dedicada à inferência, o que impôs a faixa de 1,5 a 7 bilhões de parâmetros. As chamadas de inferência foram realizadas preservando integralmente o regime *zero-shot*. Em conjunto, esses modelos materializam, em ambiente aberto e replicável, a lógica *zero-shot* inspirada no C3, permitindo avaliar de forma controlada os efeitos de idioma e de esquema sobre a qualidade do SQL gerado.

A família Gemma foi representada pela distribuição google/gemma-3-4b-pt[11], variante instrucional com aproximadamente quatro bilhões de parâmetros e vocação explícita para o português. A documentação pública de Gemma 3 registra tamanhos

[11] Disponível em: https://huggingface.co/google/gemma-3-4b-pt

oficiais e o posicionamento da linha como modelos leves e multimodais, adequados a implantação em ambientes com recursos limitados, o que atende diretamente ao cenário computacional desta pesquisa.

O OpenChat foi empregado na distribuição openchat/openchat-3.5-1210[12]. Trata-se de um modelo instrucional amplamente divulgado como tendo cerca de 7 bilhões de parâmetros, com ênfase em instruções de uso geral e em desempenho robusto em tarefas de geração. A disponibilidade de quantizações comunitárias favorece a reprodução em GPUs de memória intermediária, mantendo-o dentro da capacidade computacional do experimento sem comprometer o regime *zero-shot* adotado. Essa escolha contribui para o contraste entre uma linhagem instrucional generalista, de porte um pouco maior, e variantes menores especializadas em português ou em raciocínio.

Para a dimensão de especialização linguística em português do Brasil, adotou-se o CEIA-UFG/Gemma-3-Gaia-PT-BR-4b-it[13], resultante de pré-treinamento contínuo e ajuste instrucional sobre o google/gemma-3-4b-pt. A model card pública descreve o processo de adaptação e o foco na produção de saídas alinhadas ao PT-BR, mantendo 4B de parâmetros, o que facilita o controle de variáveis quando comparado ao Gemma original e preserva a compatibilidade com o ambiente de inferência local.

A série Qwen foi representada pelo Qwen/Qwen3-4B-Instruct-2507-FP8[14], uma variante instrucional de aproximadamente quatro bilhões de parâmetros disponibilizada em precisão FP8. A própria página do modelo destaca o perfil de inferência sugerido e as recomendações de amostragem, e a opção por FP8 foi decisiva para reduzir o consumo de memória e aumentar a vazão de tokens sem degradar a estrutura de saída, circunstância particularmente útil nos cenários de perguntas mais longas e nos esquemas com maior cardinalidade. Além de preservar a exigência de pesos abertos e execução local, o Qwen 3 reforça a dimensão multilíngue discutida na fundamentação.

Por fim, incluiu-se o deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B[15], um modelo de aproximadamente 1,5 bilhão de parâmetros obtido por destilação a partir

[12] Disponível em: https://huggingface.co/openchat/openchat-3.5-1210
[13] https://huggingface.co/CEIA-UFG/Gemma-3-Gaia-PT-BR-4b-it
[14] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507-FP8
[15] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B

da linha R1, com ênfase em raciocínio passo a passo e custo de inferência muito baixo. Essa variante funciona, no desenho experimental, como um contraponto de "porte mínimo" frente aos modelos de 4B e ao OpenChat de 7B, permitindo observar empiricamente como a capacidade paramétrica interage com restrições de prompt rígidas e com os diferentes níveis de complexidade relacional entre tabela única e esquema estrela.

O cenário de testes foi mantido constante para todas as famílias: geração de uma única instrução SQL para SQLite, sem explicações, a partir do mesmo prompt, com serialização concisa do esquema e sem qualquer amostra adicional no contexto. Avaliaram-se quatro combinações linguísticas entre pergunta e esquema (PT–PT, EN–EN, PT–EN, EN–PT), em duas estruturas de banco de dados construídas especificamente para este estudo, refletindo domínios e hierarquias da despesa pública e oferecendo contraste entre desnormalização extrema (tabela única) e desenho dimensional (esquema estrela). A adoção do regime zero-shot permite isolar a contribuição do pré-treinamento e do alinhamento instrucional de cada família, conforme discutido na fundamentação sobre *zero-shot*, *few-shot* e *fine-tuning*.

5.1 *PIPELINE* DO EXPERIMENTO

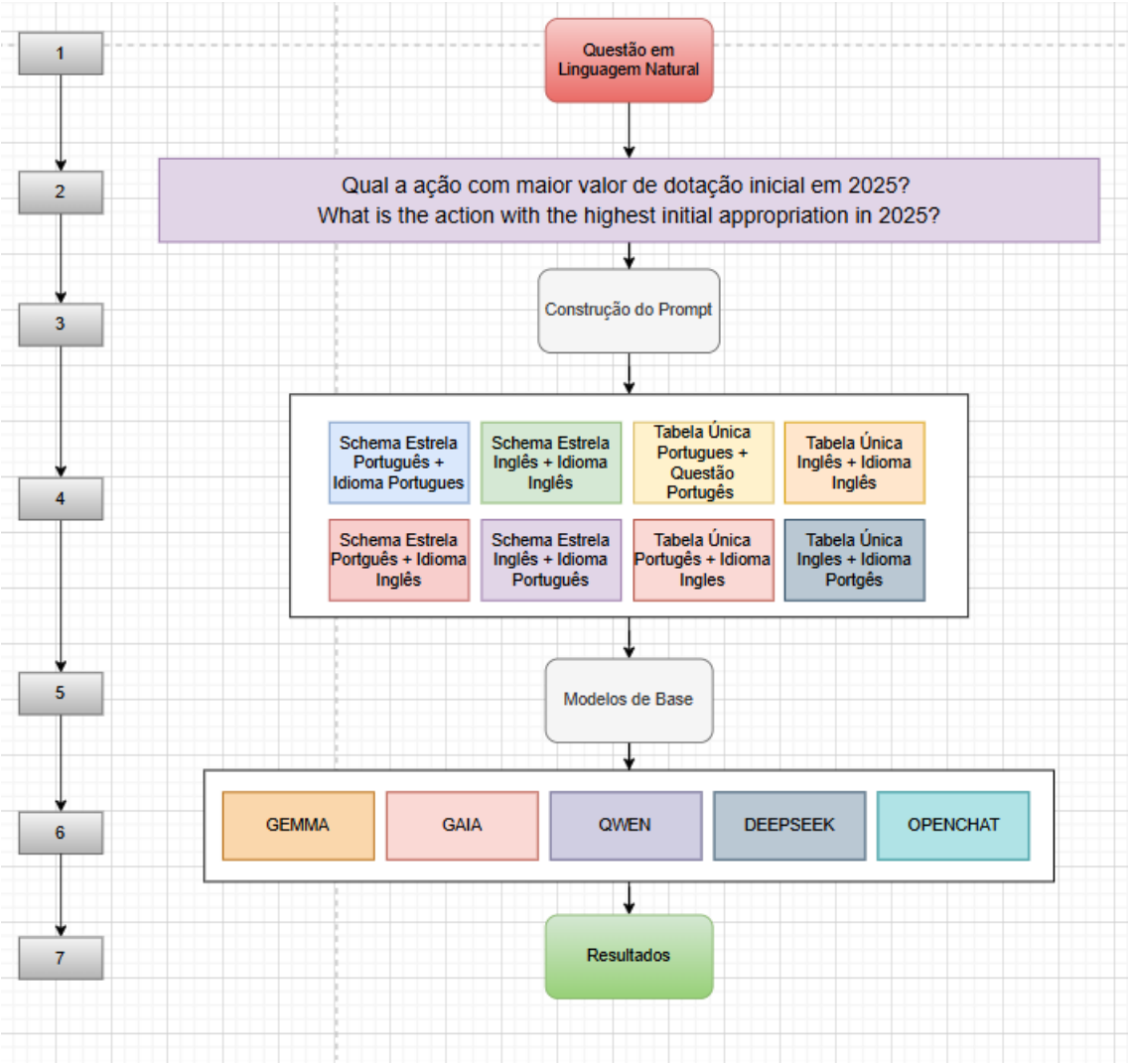
Esta seção descreve o fluxo experimental numerado na Figura 5 e organiza a apresentação de uma questão exemplificadora ao longo de todo o percurso, assegurando comparabilidade entre estruturas, idiomas e modelos.

Na etapa 1, o processo inicia com a formulação da questão em linguagem natural no domínio orçamentário. Adota-se uma redação livre acerca da base de dados, preservando a semântica entre as versões em português e em inglês quando aplicável. Nesta fase registra-se o enunciado que será utilizado no experimento, com a indicação do período de referência e do contexto administrativo. Neste experimento existe um ambiente controlado, com as questões já estabelecidas, porém neste momento quando aplicado a um ambiente real, a pessoa liberdade para discorrer acerca de qualquer questão.

Na etapa 2, a questão é estabilizada em versão bilíngue para permitir a avaliação cruzada entre idioma do enunciado e idioma do esquema. Mantém-se equivalência semântica estrita entre as duas formulações, evitando escolhas lexicais

que privilegiem um idioma em detrimento do outro. É neste ponto que se fixa a pergunta que perpassará todas as combinações subsequentes.

Figura 5 - Sequência de atividades



Fonte: Elaborado pelo Autor (2025)

Na etapa 3, procede-se à construção do prompt. O enunciado é incorporado ao molde prescritivo definido no capítulo 4.2, com ancoragem explícita no dialeto SQLite, inclusão do esquema como contexto técnico e instrução para produzir um único comando SQL válido, sem texto adicional. Nesta etapa são apresentados os *templates* finais em português e em inglês, com a interpolação da pergunta e do esquema alvo utilizados na execução.

Na etapa 4, o experimento é organizado em oito combinações resultantes do cruzamento entre modelagem de dados, idioma do esquema e idioma do enunciado. As quatro variantes de esquema — esquema estrela em português, esquema estrela em inglês, tabela única em português e tabela única em inglês — são testadas com enunciados em português e em inglês, totalizando, para cada modelagem, dois cenários alinhados (PT–PT e EN–EN) e dois cenários cruzados (PT–EN e EN–PT). Para cada uma das oito combinações, disponibiliza-se o esquema efetivo e associa-se um SQL de referência previamente validado por execução, como mencionado na seção 4.2, que funcionará como gabarito para a métrica de acurácia composta.

Na etapa 5, selecionam-se os modelos de base a serem avaliados, observando critérios de reprodutibilidade, porte compatível com execução local e disponibilidade de variantes instrucionais ou com foco em português. Todos operam em regime *zero-shot* sob as mesmas condições de inferência, garantindo comparabilidade. Na etapa 6, os modelos são aplicados às oito combinações definidas; cada modelo recebe o mesmo prompt da combinação correspondente e produz um comando SQL que é executado no banco respectivo.

Na etapa 7, consolida-se o julgamento segundo a métrica de acurácia composta. Uma saída é considerada correta apenas quando obtém simultaneamente equivalência de resultados em execução e correspondência semântica e estrutural com o SQL de referência. O registro final discrimina acertos e erros por modelo. Esse encadeamento, tal como sintetizado na Figura 5, permite acompanhar de maneira transparente o percurso da questão ao longo do pipeline, explicitando o papel do prompt, o impacto do idioma e do desenho do banco de dados e as diferenças de comportamento entre modelos em condições controladas.

5.2 DESEMPENHO NA ESTRUTURA DE TABELA ÚNICA

Na segunda configuração experimental, foi utilizada a estrutura de tabela única, na qual todos os atributos estão consolidados em uma única tabela não normalizada. Essa configuração reduz a necessidade de raciocínio relacional e simplifica a formulação das consultas SQL, o que se refletiu em um desempenho significativamente superior em todos os modelos testados.

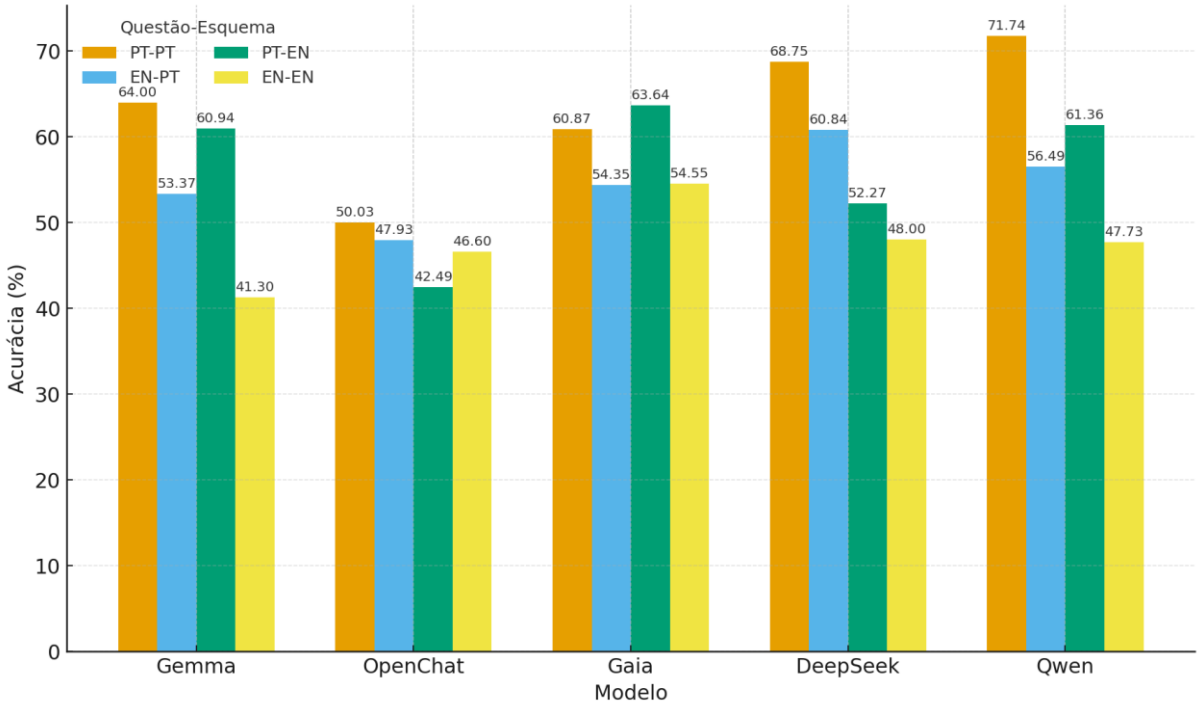
Tabela 3 - Acurácia dos modelos na tabela única.

Detentor do modelo	Modelo	Idioma da questão	Idioma do schema de dados	Acurácia (%)
google	gemma	PT	PT	64.00
google	gemma	EN	PT	53.37
google	gemma	PT	EN	60.94
google	gemma	EN	EN	41.30
open	openchat	PT	PT	50.03
open	openchat	EN	PT	47.93
open	openchat	PT	EN	42.49
open	openchat	EN	EN	46.60
ceia	gaia	PT	PT	60.87
ceia	gaia	EN	PT	54.35
ceia	gaia	PT	EN	63.64
ceia	gaia	EN	EN	54.55
deepseek	deepseek	PT	PT	68.75
deepseek	deepseek	EN	PT	60.84
deepseek	deepseek	PT	EN	52.27
deepseek	deepseek	EN	EN	48.00
alibaba	**qwen**	**PT**	**PT**	**71.74**
alibaba	qwen	EN	PT	56.49
alibaba	qwen	PT	EN	61.36
alibaba	qwen	EN	EN	47.73

Fonte: Elaborado pelo autor (2025)

Os resultados revelaram que o modelo Qwen novamente se destacou, atingindo 71,74% de acurácia na combinação PT–PT, seguido pelo DeepSeek, com 68,75%, e pelo Gemma, com 64%. O Gaia apresentou desempenho de 60,87% na mesma configuração, enquanto o OpenChat manteve valores em torno de 50%. A superioridade dos modelos Qwen e DeepSeek sugere que arquiteturas mais recentes, treinadas em grandes volumes de dados multilíngues e técnicos, são mais capazes de compreender estruturas descritivas em português e realizar a correspondência semântica adequada entre texto e atributos de banco.

Figura 6 - Resultados por modelo para a Tabela única



Fonte: Elaborado pelo autor (2025)

Além da melhora geral na acurácia, a estrutura de tabela única mostrou-se mais estável quanto à consistência sintática. O número de consultas inválidas ou não executáveis foi significativamente menor em comparação ao esquema estrela. A simplificação estrutural elimina a necessidade de raciocínio relacional e permite que os modelos se concentrem na identificação de atributos e operadores, resultando em maior precisão. Ainda assim, observou-se que as consultas que envolviam funções de agregação condicionais, como somatórios e médias filtradas, continuaram apresentando maior índice de erros lógicos, o que indica que a capacidade de raciocínio aritmético e de filtragem ainda constitui um desafio entre os modelos.

5.3 DESEMPENHO NA ESTRUTURA ESTRELA

O primeiro conjunto de experimentos considerou o modelo de dados no formato estrela, no qual a consulta correta depende da capacidade do modelo em compreender múltiplas relações entre tabelas e executar junções (joins) coerentes. Essa configuração representa um desafio mais complexo, pois exige não apenas o reconhecimento semântico da pergunta, mas também o raciocínio sobre chaves primárias, estrangeiras e relacionamentos.

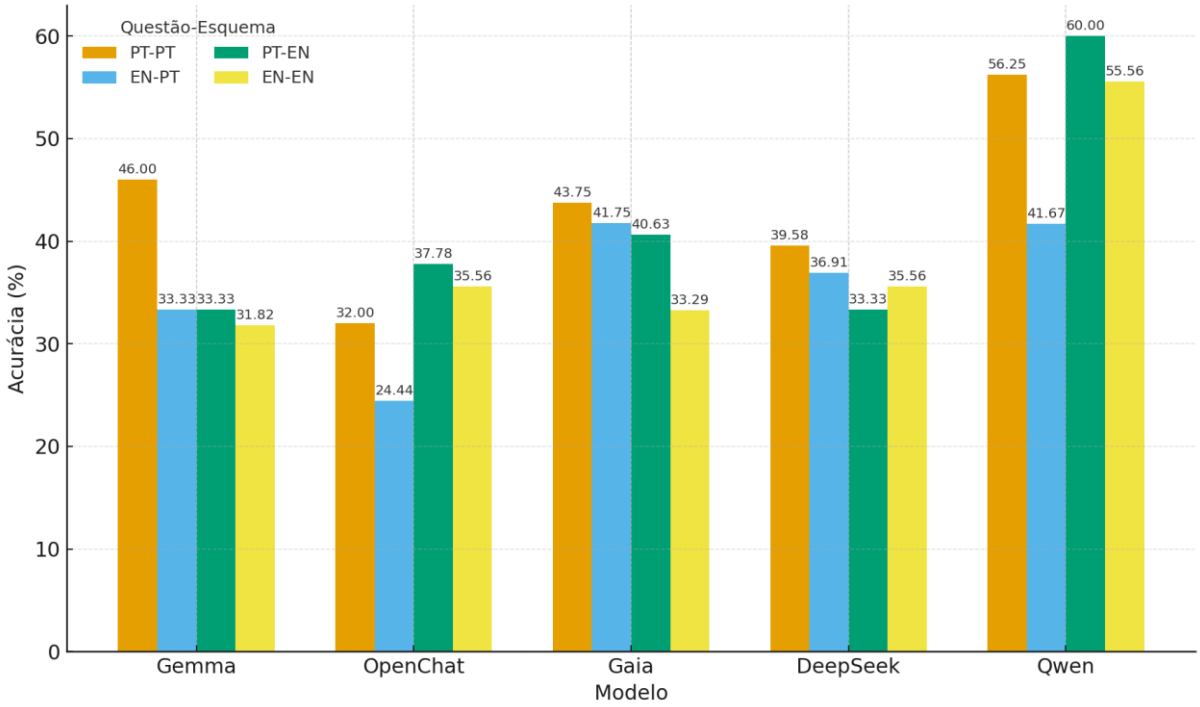
Tabela 4 - Acurácia dos Modelos no formato Star Schema

Detentor do modelo	Modelo	Idioma da questão	Idioma do schema de dados	Acurácia (%)
google	gemma	PT	PT	46.00
google	gemma	EN	PT	33.33
google	gemma	PT	EN	33.33
google	gemma	EN	EN	31.82
open	openchat	PT	PT	32.00
open	openchat	EN	PT	24.44
open	openchat	PT	EN	37.78
open	openchat	EN	EN	35.56
ceia	gaia	PT	PT	43.75
ceia	gaia	EN	PT	41.75
ceia	gaia	PT	EN	40.63
ceia	gaia	EN	EN	33.29
deepseek	deepseek	PT	PT	39.58
deepseek	deepseek	EN	PT	36.91
deepseek	deepseek	PT	EN	33.33
deepseek	deepseek	EN	EN	35.56
alibaba	qwen	PT	PT	56.25
alibaba	qwen	EN	PT	41.67
alibaba	**qwen**	**PT**	**EN**	**60.00**
alibaba	qwen	EN	EN	55.56

Fonte: Elaborado pelo autor (2025)

Os resultados obtidos demonstraram diferenças significativas entre os modelos avaliados. O Qwen, desenvolvido pela Alibaba, apresentou o melhor desempenho geral, atingindo 56,25% de acurácia na configuração em que tanto a pergunta quanto o *schema* estavam em português (PT–PT) e 60,00% na combinação PT–EN. Esses resultados indicam uma alta capacidade de adaptação multilíngue, além de um desempenho consistente mesmo quando a linguagem da pergunta e do *schema* não coincidem. O Gemma 3B, da Google, apresentou comportamento estável, com desempenho médio entre 31% e 46%, enquanto o modelo nacional Gaia, desenvolvido pelo CEIA/UFG, obteve resultados próximos a 44% na condição PT–PT. O DeepSeek, modelo de origem chinesa, demonstrou acurácia intermediária (cerca de 40% em PT–PT), mantendo coerência semântica nas consultas geradas, ainda que com maior propensão a erros de estrutura em junções complexas. O OpenChat, baseado em LLaMA, apresentou os resultados mais modestos, com valores variando entre 24% e 38% de acurácia, demonstrando limitações na manipulação de múltiplas tabelas e nas cláusulas de agrupamento.

Figura 7 - Resultados por modelo para o Esquema Estrela



Fonte: Elaborado pelo autor (2025)

De modo geral, verificou-se que o desempenho dos modelos é afetado pela complexidade estrutural do esquema estrela. A presença de múltiplas tabelas e relacionamentos aumenta substancialmente o risco de erros estruturais e sintáticos. Em diversos casos, observou-se que os modelos geravam comandos SQL semanticamente coerentes, mas que não eram executáveis devido a erros em nomes de colunas ou em junções incorretas. Essa dificuldade foi mais evidente nas configurações em que o idioma da pergunta diferia do idioma do *schema*, evidenciando que a tradução implícita de termos técnicos, como “orgão”, “função” ou “subfunção”, ainda representa um desafio considerável para os LLMs em contextos bilíngues.

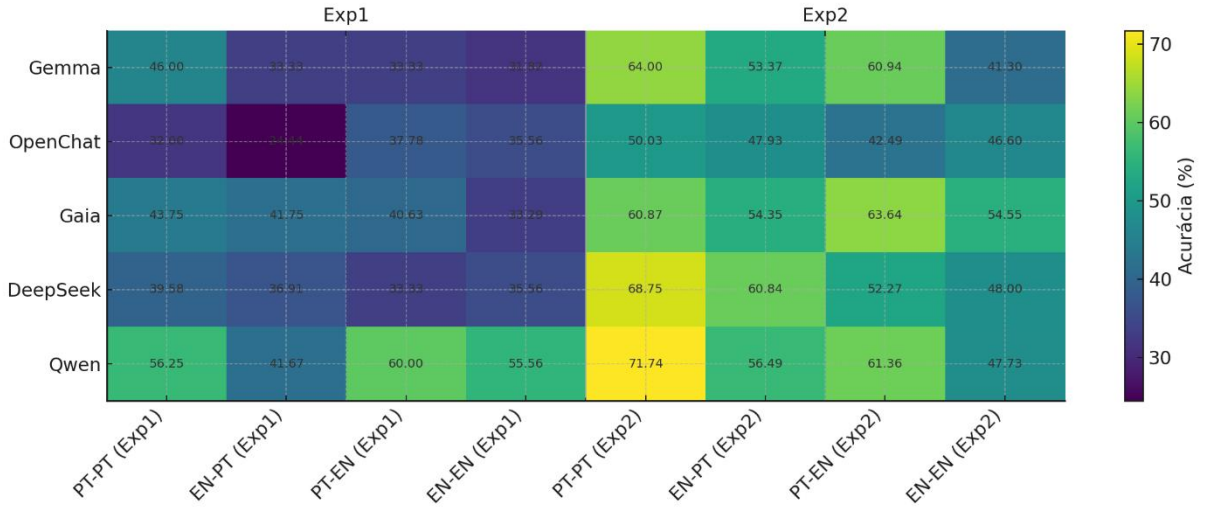
5.4 DESEMPENHO GERAL

De modo consolidado, os resultados confirmam uma tendência: quanto menor a complexidade estrutural do esquema de dados e maior a correspondência linguística entre a pergunta e o *schema*, maior é a acurácia dos modelos. Como sintetizado no mapa de calor da Figura 8, o Qwen destaca-se como o melhor desempenho global

nas duas baterias de experimentos, seguido pelo DeepSeek. Essa hierarquia sugere vantagem de arquiteturas recentes treinadas com foco em instruções multilíngues e tarefas próximas à geração de código, o que favorece a tradução semântica entre linguagem natural e SQL. O Gemma 3B apresentou resultados consistentes, porém abaixo dos líderes, enquanto o Gaia manteve competitividade notável em cenários com perguntas em português. O OpenChat foi sistematicamente inferior nas quatro combinações de idiomas, sobretudo em configurações que demandam operações relacionais mais complexas. Entre os dois experimentos observou-se ganho generalizado de acurácia na maioria das células do mapa, com aumentos absolutos que chegam a aproximadamente 29 pontos percentuais no caso do DeepSeek em PT-PT, embora haja exceções pontuais, como a queda do Qwen em EN-EN. Em termos de pareamento linguístico, a combinação PT-PT produziu os maiores índices médios de acerto, o que indica que a uniformidade de idioma reduz ambiguidades de mapeamento entre termos da pergunta e atributos do banco. O Qwen manteve desempenho relativamente mais equilibrado nas configurações bilíngues, sinalizando melhor capacidade de generalização entre contextos linguísticos distintos.

Os resultados reforçam que a natureza do esquema exerce influência direta sobre a geração de SQL. Estruturas não normalizadas, como a tabela única, reduzem o espaço de busca e o número de dependências relacionais, concentrando o esforço do modelo no alinhamento semântico entre rótulos de atributos e termos da pergunta. Em contraste, o modelo estrela requer a compreensão de hierarquias e relacionamentos que ainda não são plenamente capturados pelos modelos avaliados em regime *zero-shot*, mesmo quando treinados extensivamente com dados de código e instruções complexas. As diferenças entre combinações de idioma, refletidas no gradiente de cores da Figura 8, mostram que a coerência lexical entre corpus de treinamento implícito e vocabulário da tarefa permanece um determinante central de desempenho, e que modelos com treinamento intensivo em tradução e raciocínio multilingue tendem a sofrer menos degradação ao alternar entre perguntas e *schemas* em idiomas distintos.

Figura 8 - Resultados consolidados por modelo



Fonte: Elaborado pelo autor (2025)

Os resultados desta pesquisa, obtidos a partir de uma base desenvolvida de despesas públicas estruturada em dois arranjos complementares de dados, um em modelo estrela e outro em tabela única, avaliados em variações bilíngues de *schema* e de enunciado, demonstram de forma clara o impacto simultâneo da complexidade estrutural e da correspondência linguística sobre a geração de SQL em regime *zero-shot*. Com dois *prompts* criados para reduzir ambiguidade e explicitar a tarefa sem induzir soluções, e mantendo condições controladas de inferência, observou-se desempenho consolidado entre 24,44% e 71,74% de acurácia, com supremacia do Qwen nas quatro combinações de idiomas e pico em PT–PT, avanços expressivos do DeepSeek entre os experimentos, consistência do Gemma e competitividade do Gaia em português, enquanto o OpenChat permaneceu inferior nas demandas relacionais. Esses achados confirmam que a desnormalização mitiga o espaço de busca e que o pareamento idiomático favorece o mapeamento semântico entre termos e atributos, oferecendo evidência empírica reprodutível de como escolhas de modelagem de dados e de formulação de *prompts* modulam o desempenho de LLMs em *Text-To-SQL*. A construção da base bilíngue, a orquestração dos experimentos, a análise comparativa entre arquiteturas e a sistematização dos ganhos entre condições constituem uma contribuição original e aplicada, suficiente para sustentar os objetivos propostos.

CONCLUSÃO

A dissertação investigou, em cenário *zero-shot*, a capacidade de modelos de linguagem de grande escala em traduzir perguntas em linguagem natural para consultas SQL executáveis sobre dados públicos. Para isso, foi concebido um ambiente de avaliação reprodutível composto por uma base bilíngue representativa do domínio orçamentário de Santa Catarina, implementada em oito variantes que combinam tabela única e esquema estrela, em português e em inglês, e por um conjunto de cinquenta perguntas com respectivos SQL de referência, traduzidos para ambos os idiomas. O protocolo experimental manteve constante um molde de prompt prescritivo e adotou uma métrica única de acurácia composta, segundo a qual uma saída é correta somente quando a execução retorna o mesmo resultado do gabarito.

Os resultados indicaram que decisões de modelagem e de idioma afetam de maneira substantiva o desempenho. A representação em tabela única apresentou, em média, acurácia superior à do esquema estrela, sugerindo que a redução do custo de junções e a exposição direta de rótulos favorecem a tradução semântica em regime zero-shot. Observou-se ainda um efeito positivo, embora não absoluto, do alinhamento entre o idioma do enunciado e o do esquema, com ganhos mais visíveis em consultas que envolvem agregações e filtros temporais. Entre os modelos avaliados, o Qwen obteve o melhor desempenho global nas combinações testadas, alcançando os maiores valores de acurácia e superando as demais alternativas de porte semelhante. Esses achados reforçam a hipótese de que, na ausência de adaptação supervisionada, a escolha de um desenho de dados mais simples e de um alinhamento linguístico cuidadoso pode ser tão decisiva quanto a arquitetura do modelo.

Do ponto de vista metodológico, a principal contribuição reside na disponibilização de um recurso experimental bilíngue, com esquemas paralelos e gabaritos validados por execução, e em um protocolo de avaliação compatível com execução local e com restrições realistas de infraestrutura. Em contraste com *benchmarks* generalistas voltados ao inglês, o material produzido reflete as especificidades da administração pública brasileira e permite mensurar, com controle de variáveis, os efeitos de idioma e de modelagem na tarefa de *Text-To-SQL* em português. O estudo também oferece evidências úteis para órgãos públicos e equipes técnicas: quando o objetivo é facilitar o acesso automatizado por interfaces em linguagem natural, publicar visões não

normalizadas ou tabelas consolidadas, acompanhadas de catálogos bilíngues consistentes, tende a reduzir erros de ligação de esquema e a aumentar a taxa de respostas corretas.

Algumas limitações devem ser reconhecidas. A avaliação concentrou-se em um único domínio setorial e em um recorte temporal específico, com um conjunto de cinquenta perguntas desenhado para cobrir operações recorrentes no contexto orçamentário. Os experimentos consideraram apenas modelos abertos de porte reduzido, executados sem ajuste fino, e utilizaram SQLite como dialeto de referência, o que pode afetar a generalização para outros sistemas gerenciadores.

Essas restrições abrem frentes claras de continuidade. Trabalhos futuros podem ampliar o conjunto de perguntas e de domínios, estender a avaliação para outros estados e órgãos, e explorar variantes de *prompting* com exemplos mínimos, recuperação de metadados e gramáticas de saída que restrinjam a geração ao espaço de SQL válido. Estratégias semi-supervisionadas e de pseudo-rotulagem por execução, combinadas a dados públicos adicionais, constituem caminhos promissores para adaptar modelos ao contexto brasileiro com baixo custo de anotação. A comparação controlada entre diferentes dialetos e a incorporação de amostras de linhas no contexto podem esclarecer o papel de pistas distribuídas do próprio dado na qualidade da tradução.

Em síntese, a dissertação demonstra que é viável avaliar, com rigor e reprodutibilidade, o *Text-To-SQL* em português sobre dados públicos reais, fornecendo artefatos e evidências que auxiliam tanto a pesquisa acadêmica quanto a prática institucional. Ao mostrar que escolhas de modelagem e de idioma impactam de forma mensurável a acurácia em *zero-shot* e ao identificar modelos mais adequados ao cenário regional, o estudo contribui para reduzir barreiras de acesso e para orientar iniciativas que buscam tornar dados governamentais mais utilizáveis por meio de interfaces em linguagem natural.

REFERÊNCIAS

COMRIE, B. **Language universals and linguistic typology: syntax and morphology**. Chicago: University of Chicago Press, 1989.

DENG, N.; CHEN, Y.; ZHANG, Y. Recent advances in Text-to-SQL: a survey of what we have and what we expect. In: **Proceedings of the 29th International Conference on Computational Linguistics**, 2022. p. 2166–2187.

BUCHINGER, D.; CAVALCANTI, G. A. d. S.; HOUNSELL, M. D. S. Mecanismos de busca acadêmica: uma análise quantitativa. **Revista Brasileira de Computação Aplicada**, v. 6, n. 1, 2014.

JOSÉ, M. M. et al. Integrating question answering and text-to-SQL in Portuguese. In: **Conference on Computational Linguistics in Brazil**. 2022. p. 123–136.

JOSÉ, M. A.; COZMAN, F. G. mRAT-SQL+GAP: a Portuguese text-to-SQL transformer. In: **International Conference on Computational Linguistics**. 2021. p. 567–580.

GUO, J. et al. Chase: a large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL. In: **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Online: Association for Computational Linguistics, 2021. p. 2316–2331.

TALAEI, S. et al. Chess: contextual harnessing for efficient SQL synthesis. arXiv preprint arXiv:2405.16755, 2024. Disponível em: https://arxiv.org/abs/2405.16755 Acesso em: 12 out. 2024.

ZHONG, V.; XIONG, C.; SOCHER, R. Seq2SQL: generating structured queries from natural language using reinforcement learning. **CoRR**, v. abs/1709.00103, 2017. Disponível em: http://arxiv.org/abs/1709.00103. Acesso em: 12 out. 2024.

YU, T. et al. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)**. Brussels: Association for Computational Linguistics, 2018. p. 3911–3921. Disponível em: https://doi.org/10.18653/v1/D18-1425. Acesso em: 15 out. 2024.

YU, T. et al. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)**, Brussels, Belgium, 31 out.–4 nov. 2018. Association for Computational Linguistics, 2018. p. 3911–3921. Disponível em: https://doi.org/10.18653/v1/D18-1425. Acesso em: 15 out. 2024.

LI, J. et al. Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. In: **Advances in Neural Information Processing Systems 36 (NeurIPS 2023)**, New Orleans, LA, 10–16 dez. 2023. Disponível em: http://papers.nips.cc/paperfiles/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-DatasetsandBenchmarks.html. Acesso em: 20 out. 2024.

LI, B. et al. The dawn of natural language to SQL: are we fully ready? arXiv preprint arXiv:2406.01265, 2024. Disponível em: https://arxiv.org/abs/2406.01265. Acesso em: 20 out. 2024.

GAO, D. et al. **Text-to-SQL empowered by large language models: a benchmark evaluation**. arXiv preprint arXiv:2308.15363, 2023. Disponível em: https://arxiv.org/abs/2308.15363. Acesso em: 19 out. 2024.

DONG, X.; ZHANG, C.; GE, Y. **C3: zero-shot Text-to-SQL with ChatGPT**. arXiv preprint arXiv:2307.07306, 2023. Disponível em: https://arxiv.org/abs/2307.07306. Acesso em: 20 out. 2024.

GARCIA, G. L. et al. **Introducing Bode: a fine-tuned large language model for Portuguese prompt-based task**. arXiv preprint arXiv:2401.02909, 2024. Disponível em: https://arxiv.org/abs/2401.02909. Acesso em: 15 nov. 2024.

LI, J. et al. **Can LLM already serve as a database interface? A big bench for large-scale database grounded Text-to-SQLs**. arXiv preprint arXiv:2305.08872, 2023. Disponível em: https://arxiv.org/abs/2305.03111. Acesso em: 10 jan. 2025.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)**. [S.l.: s.n.], 2008. p. 68–77.

VASWANI, A. et al. Attention is all you need. In: **Advances in Neural Information Processing Systems 30 (NeurIPS 2017)**. Long Beach, CA: Curran Associates, 2017. p. 5998–6008.

HENDRIX, G. G.; SACERDOTI, E. D. Natural-language processing: the field in perspective. **Byte**, v. 6, n. 8, p. 304–352, 1981.

LI, F.; JAGADISH, H. V. Constructing an interactive natural language interface for relational databases. **Proceedings of the VLDB Endowment**, v. 8, n. 1, p. 73–84, 2014.

DAHL, D. A. Expanding the scope of the ATIS task: the ATIS-3 corpus. In: **Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8–11, 1994**. [S.l.: s.n.], 1994. Disponível em: https://aclanthology.org/H94-1010. Acesso em: 22 dez. 2024.

ZELLE, J. M.; MOONEY, R. J. Learning to parse database queries using inductive logic programming. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. Portland, Oregon: The AAAI Press, 1996. v. 13. ISBN 978-0-262-51091-2.

ZHONG, V.; XIONG, C.; SOCHER, R. Seq2SQL: generating structured queries from natural language using reinforcement learning. **CoRR**, v. abs/1709.00103, 2017. Disponível em: http://arxiv.org/abs/1709.00103. Acesso em: 26 dez. 2024.

GONZALEZ, M.; LIMA, V. L. S. Recuperação de informação e processamento da linguagem natural. In: **Congresso da Sociedade Brasileira de Computação, 23., 2003, Campinas**. Anais [...]. Porto Alegre: SBC, 2003. p. 347–395. Disponível em: https://www.marilia.unesp.br/Home/Instituicao/Docentes/EdbertoFerneda/mri-06---gonzales-e-lima-2003.pdf. Acesso em: 28 dez. 2024.

BATES, M. **Models of natural language understanding human-machine communication by voice**. Irvine: National Academy of Sciences, 1993.

ARAMPATZIS, A. T.; VAN DER WEIDE, T. P.; KOSTER, C. H. A.; VAN BOMMEL, P. Linguistically-motivated information retrieval. In: **Encyclopedia of Library and Information Science**, v. 69, 2000. p. 201–222.

SILVA, B. et al. **Introdução ao processamento das línguas naturais e algumas aplicações**. São Carlos: NILC-ICMC-USP, 2007. (Série de Relatórios do Núcleo

Interinstitucional de Linguística Computacional, NILC-TR-07-10). Disponível em: https://sites.icmc.usp.br/taspardo/nilctr0710-diasdasilvaetal.pdf. Acesso em: 28 dez. 2024.

JACQUEMIN, C.; KLAVANS, J. L.; TZOUKERMANN, E. Expansion of multi-word terms for indexing and retrieval using morphology and syntax. In: **35th Annual Meeting of the ACL – 8th Conference of the European Chapter of the ACL**, Madri, 1997. p. 24–31.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Estudos Avançados**, v. 35, p. 85–94, 2021.

MITCHELL, T. **Machine learning**. S.l.: McGraw Hill, 1997.

PAULA, G. A. **Modelos de regressão: com apoio computacional**. São Paulo: IME-USP, 2004. p. 28–55.

SILVA, S. F. da. **Seleção de características por meio de algoritmos genéticos para aprimoramento de rankings e de modelos de classificação**. 2011. Tese (Doutorado em Ciência da Computação) – Universidade de São Paulo, São Paulo, 2011.

FLECK, L.; TAVARES, M. H. F.; EYNG, E.; HELMANN, A. C.; ANDRADE, M. A. D. M. Redes neurais artificiais: princípios básicos. **Revista Eletrônica Científica Inovação e Tecnologia**, v. 1, n. 13, p. 47–57, 2016.

ELMASRI, R.; NAVATHE, S. B. [Trad.]. **Sistemas de bancos de dados**. Traduzido do original: *Fundamentals of Database Systems*. São Paulo: Pearson (Addison Wesley), 2005. 724 p. ISBN 85-88639-17-3.

HOLDSWORTH, J.; STRYKER, C. **What is natural language processing?** Disponível em: https://www.ibm.com/topics/natural-language-processing. Acesso em: 11 jan. 2025.

RAMOS, A. S. M. **Inteligência artificial generativa baseada em grandes modelos de linguagem: ferramentas de uso na pesquisa acadêmica**. In: **SciELO Preprints**, 2023. Disponível em: https://doi.org/10.1590/SciELOPreprints.6105. Acesso em: 11 jan. 2025.

RAFFEL, C. et al. Exploring the limits of transfer learning with a unified text-to-text transformer. **Journal of Machine Learning Research**, v. 21, n. 140, p. 1–67, 2020.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding by generative pre-training**. OpenAI, 2018. Disponível em: https://openai.com/research/. Acesso em: 11 jan. 2025.

YANG, Z. et al. **XLNet: generalized autoregressive pretraining for language understanding**. Carnegie Mellon University; Google AI Brain Team, 2019. Disponível em: https://arxiv.org/abs/1906.08237. Acesso em: 11 jan. 2025.

LIU, Y. et al. **RoBERTa: a robustly optimized BERT pretraining approach**. University of Washington; Facebook AI, 2019. Disponível em: https://arxiv.org/abs/1907.11692. Acesso em: 11 jan. 2025.

BOMMASANI, R. et al. **On the opportunities and risks of foundation models**. Stanford University, 2021. Disponível em: https://arxiv.org/abs/2108.07258. Acesso em: 12 jan. 2025.

THIRUNAVUKARASU, A. J. et al. Large language models in medicine. **Nature Medicine**, 2023, p. 1–11.

BRANTS, T. et al. **Large language models in machine translation**. 2023. Disponível em: https://aclanthology.org/D07-1090.pdf. Acesso em: 12 jan. 2025.

KASNECI, E. et al. **ChatGPT for good? On opportunities and challenges of large language models for education**. 2023. Disponível em: https://doi.org/10.1016/j.lindif.2023.102274. Acesso em: 12 jan. 2025.

YANG, H. et al. **Large language models meet text-centric multimodal sentiment analysis: a survey**. 2024. Disponível em: https://arxiv.org/pdf/2406.08068. Acesso em: 12 jan. 2025.

SU, Y. et al. **Automation and machine learning augmented by large language models in a catalysis study**. 2024. Disponível em: https://doi.org/10.1039/D3SC07012C. Acesso em: 12 jan. 2025.

LAI, J.; GAN, W.; WU, J.; QI, Z.; YU, P. S. **Large language models in law: a survey**. 2024. Disponível em: https://doi.org/10.1016/j.aiopen.2024.09.002. Acesso em: 12 jan. 2025.

SHI, L.; TANG, Z.; ZHANG, N.; ZHANG, X.; YANG, Z. **A survey on employing large language models for Text-to-SQL tasks**. 2024. Disponível em: https://arxiv.org/pdf/2407.15186. Acesso em: 12 jan. 2025.

LI, J. et al. Can LLM already serve as a database interface? A big bench for large-scale database grounded Text-to-SQLs. In: **Advances in Neural Information Processing Systems**, v. 36, 2024.

RAI, D.; WANG, B.; ZHOU, Y.; YAO, Z. **Improving generalization in language model-based Text-to-SQL semantic parsing: two simple semantic boundary-based techniques**. 2023. Disponível em: https://arxiv.org/pdf/2305.17378. Acesso em: 12 jan. 2025.

CHEN, Z. et al. **ShadowGNN: graph projection neural network for Text-to-SQL parser**. arXiv preprint arXiv:2104.04689, 2021. Disponível em: https://arxiv.org/pdf/2104.04689. Acesso em: 12 jan. 2025.

LIN, X. V.; SOCHER, R.; XIONG, C. **Bridging textual and tabular data for cross-domain Text-to-SQL semantic parsing**. arXiv preprint arXiv:2012.12627, 2020. Disponível em: https://arxiv.org/pdf/2012.12627. Acesso em: 12 jan. 2025.

CAO, R. et al. **LGESQL: line graph enhanced Text-to-SQL model with mixed local and non-local relations**. arXiv preprint arXiv:2106.01093, 2021. Disponível em: https://arxiv.org/pdf/2106.01093. Acesso em: 12 jan. 2025.

XU, P. et al. **Optimizing deeper transformers on small datasets**. arXiv preprint arXiv:2012.15355, 2021. Disponível em: https://arxiv.org/pdf/2012.15355. Acesso em: 12 jan. 2025.

GAO, D. et al. **Text-to-SQL empowered by large language models: a benchmark evaluation**. arXiv preprint arXiv:2308.15363, 2023. Disponível em: https://arxiv.org/pdf/2308.15363. Acesso em: 12 jan. 2025.

SPECIA, L. **Uma abordagem híbrida relacional para a desambiguação lexical de sentido na tradução automática**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo, [S.l.], 2007.

ZHONG, R.; YU, T.; KLEIN, D. **Semantic evaluation for Text-to-SQL with distilled test suites**. arXiv preprint arXiv:2010.02840, 2020. Disponível em: https://arxiv.org/abs/2010.02840. Acesso em: 12 jan. 2025.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding by generative pre-training**. San Francisco: OpenAI, 2018.

Disponível em: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. Acesso em: 26 out. 2025.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)**, Minneapolis. Stroudsburg, PA: Association for Computational Linguistics, 2019. p. 4171–4186. DOI: 10.18653/v1/N19-1423.

VASWANI, A. et al. Attention is all you need. In: **Advances in Neural Information Processing Systems 30 (NeurIPS 2017)**, Long Beach, CA. New York: Curran Associates, 2017. p. 5998–6008. Disponível em: https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf. Acesso em: 26 out. 2025.

YAROWSKY, D. Unsupervised word sense disambiguation rivaling supervised methods. In: **Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL 1995)**, Cambridge, MA. Stroudsburg, PA: Association for Computational Linguistics, 1995. p. 189–196. DOI: 10.3115/981658.981684.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: **COLT'98: Proceedings of the Eleventh Annual Conference on Computational Learning Theory**, Madison, WI. New York: ACM, 1998. p. 92–100. DOI: 10.1145/279943.279962.

TARVAINEN, A.; VALPOLA, H. **Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results**. arXiv preprint arXiv:1703.01780, 2017. Disponível em: https://arxiv.org/abs/1703.01780. Acesso em: 26 out. 2025.

BERTHELOT, D. et al. MixMatch: a holistic approach to semi-supervised learning. In: **Advances in Neural Information Processing Systems 32 (NeurIPS 2019)**, Vancouver. New York: Curran Associates, 2019. DOI: 10.5555/3454287.3454741.

SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural machine translation of rare words with subword units. In: **Proceedings of the 54th Annual Meeting of the Association**

for Computational Linguistics (ACL 2016), Berlin. Stroudsburg, PA: Association for Computational Linguistics, 2016. p. 1715–1725. DOI: 10.18653/v1/P16-1162.

BRASIL. Lei nº 12.527, de 18 de novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5º da Constituição Federal [...]. Diário Oficial da União, Brasília, DF, 18 nov. 2011.

APENDICE A – CÓDIGO FONTE DA TRADUÇÃO DAS QUESTÕES

```
from datasets import load_dataset
import asyncio
import pandas as pd
from concurrent.futures import ThreadPoolExecutor
from deep_translator import GoogleTranslator
from concurrent.futures import ThreadPoolExecutor
import time
from tqdm import tqdm

ds = load_dataset("VictorDCh/spider-clean-Text-To-SQL-3")

# Converte cada parte para DataFrame do pandas
train_df = pd.DataFrame(ds['train'])
print(len(train_df))
dev_df = pd.DataFrame(ds['dev'])
print(len(dev_df))
test_df = pd.DataFrame(ds['test'])
print(len(test_df))

#train_df.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/train_df.csv')
#dev_df.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/dev_df.csv')
test_df.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/test_df.csv')

def traduzir_batch(textos, idioma_destino):
    """
    Traduz um lote de textos para o idioma de destino.
    """
    try:
        return [GoogleTranslator(target=idioma_destino).translate(texto) for texto in textos]
    except Exception as e:
        print(f"Erro ao traduzir lote: {e}")
        return [None] * len(textos)

def traduzir_coluna_em_lotes(df, coluna_origem, coluna_destino, idioma_destino, tamanho_lote=100,
num_threads=10):
    """
    Traduz uma coluna inteira de um DataFrame em lotes e usando paralelismo.
    """
```

```
    textos = df[coluna_origem].tolist()
    resultados = []

    # Divide os textos em lotes
    lotes = [textos[i:i + tamanho_lote] for i in range(0, len(textos), tamanho_lote)]

    # Inicializa a barra de progresso
    total_lotes = len(lotes)
    start_time = time.time()

    with ThreadPoolExecutor(max_workers=num_threads) as executor:
        futures = [executor.submit(traduzir_batch, lote, idioma_destino) for lote in lotes]
        for i, future in enumerate(futures):
            try:
                resultados.extend(future.result())
            except Exception as e:
                print(f"Erro ao processar futuro: {e}")
                resultados.extend([None] * tamanho_lote)

            # Atualiza a barra de progresso
            elapsed_time = time.time() - start_time
            remaining_time = (elapsed_time / (i + 1)) * (total_lotes - (i + 1))
            tqdm.write(f"Processando lote {i + 1}/{total_lotes}. Tempo restante estimado:
{remaining_time:.2f} segundos.")

    # Adiciona os resultados ao DataFrame
    df[coluna_destino] = resultados
    return df


print("Iniciando tradução...")

# Traduzir a coluna para o português
#train_df_traduzido = traduzir_coluna_em_lotes(train_df, "question", "questão_traduzida", "pt")
#dev_df_traduzido = traduzir_coluna_em_lotes(dev_df, "question", "questão_traduzida", "pt")
test_df_traduzido = traduzir_coluna_em_lotes(test_df, "question", "questão_traduzida", "pt")

print("Tradução concluída.")

#train_df_traduzido.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/train_df_traduzido.c
```

sv')
#dev_df_traduzido.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/dev_df_traduzido.csv')
test_df_traduzido.to_csv(path_or_buf='/home/jonas/bode/data/questions_spider/test_df_traduzido.csv')

APENDICE B – DDL TABELA ÚNICA EM PORTUGUÊS

```
CREATE TABLE fato_despesas (
    ano TEXT,
    nro_mes TEXT,
    mes TEXT,
    nro_bimestre TEXT,
    bimestre TEXT,
    nro_trimestre TEXT,
    trimestre TEXT,
    nro_quadrimestre TEXT,
    quadrimestre TEXT,
    nro_semestre TEXT,
    semestre TEXT,
    cod_poder TEXT,
    poder TEXT,
    cod_orgao TEXT,
    orgao TEXT,
    cod_ug TEXT,
    unidade_gestora TEXT,
    cod_gestao TEXT,
    gestao TEXT,
    cod_tipo_entidade TEXT,
    tipo_entidade TEXT,
    cod_funcao TEXT,
    funcao TEXT,
    cod_subfuncao TEXT,
    subfuncao TEXT,
    cod_programa TEXT,
    programa TEXT,
    cod_acao TEXT,
    acao TEXT,
    cod_subacao TEXT,
    subacao TEXT,
    cod_uso TEXT,
    uso TEXT,
    cod_fonte TEXT,
    fonte TEXT,
    cod_grupo TEXT,
    grupo TEXT,
```

```
    cod_especificacao TEXT,
    especificacao TEXT,
    cod_tipo TEXT,
    tipo TEXT,
    cod_categoria TEXT,
    categoria TEXT,
    cod_grupo_despesa TEXT,
    grupo_despesa TEXT,
    cod_modalidade TEXT,
    modalidade TEXT,
    cod_elemento TEXT,
    elemento TEXT,
    cod_subelemento TEXT,
    subelemento TEXT,
    cod_credor TEXT,
    credor TEXT,
    indicador_emergencial TEXT,
    descricao_emergencial TEXT,
    vl_dotacao_inicial REAL,
    vl_dotacao_atualizada REAL,
    vl_empenhado REAL,
    vl_liquidado REAL,
    vl_pago_orcamentario REAL
)
```

APENDICE C – DDL TABELA ÚNICA EM INGLÊS

```
CREATE TABLE expense_fact (
    year TEXT,
    month_number TEXT,
    month TEXT,
    bimester_number TEXT,
    bimester TEXT,
    trimester_number TEXT,
    trimester TEXT,
    quadrimester_number TEXT,
    quadrimester TEXT,
    semester_number TEXT,
    semester TEXT,
    power_code TEXT,
    power_name TEXT,
    agency_code TEXT,
    agency_name TEXT,
    unit_code TEXT,
    management_unit TEXT,
    management_code TEXT,
    management_name TEXT,
    entity_type_code TEXT,
    entity_type TEXT,
    function_code TEXT,
    function_name TEXT,
    subfunction_code TEXT,
    subfunction_name TEXT,
    program_code TEXT,
    program_name TEXT,
    action_code TEXT,
    action_name TEXT,
    subaction_code TEXT,
    subaction_name TEXT,
    usage_code TEXT,
    usage_name TEXT,
    source_code TEXT,
    source_name TEXT,
    group_code TEXT,
    group_name TEXT,
```

```
    specification_code TEXT,
    specification_name TEXT,
    type_code TEXT,
    type_name TEXT,
    category_code TEXT,
    category_name TEXT,
    expense_group_code TEXT,
    expense_group_name TEXT,
    modality_code TEXT,
    modality_name TEXT,
    element_code TEXT,
    element_name TEXT,
    subelement_code TEXT,
    subelement_name TEXT,
    creditor_code TEXT,
    creditor_name TEXT,
    emergency_indicator TEXT,
    emergency_description TEXT,
    initial_budget_value REAL,
    updated_budget_value REAL,
    committed_value REAL,
    settled_value REAL,
    paid_budget_value REAL
)
```

APENDICE D – DDL STAR SCHEMA EM PORTUGUÊS

```
-- acao definição
CREATE TABLE acao (
    cod_acao TEXT PRIMARY KEY,
    nome_acao TEXT
);

-- categoria_economica definição
CREATE TABLE categoria_economica (
    cod_categoria TEXT PRIMARY KEY,
    nome_categoria TEXT
);

-- elemento definição
CREATE TABLE elemento (
    cod_elemento TEXT PRIMARY KEY,
    nome_elemento TEXT
);

-- especificacao_fonte definição
CREATE TABLE especificacao_fonte (
    cod_especificacao TEXT PRIMARY KEY,
    nome_especificacao TEXT
);

-- fonte_recurso definição
CREATE TABLE fonte_recurso (
    cod_fonte TEXT PRIMARY KEY,
    nome_fonte TEXT
);

-- funcao definição
CREATE TABLE funcao (
    cod_funcao TEXT PRIMARY KEY,
    nome_funcao TEXT
);

-- gestao definição
CREATE TABLE gestao (
```

```
    cod_gestao TEXT PRIMARY KEY,
    nome_gestao TEXT
);

-- grupo_despesa definição
CREATE TABLE grupo_despesa (
    cod_grupo TEXT PRIMARY KEY,
    nome_grupo TEXT
);

-- grupo_fonte definição
CREATE TABLE grupo_fonte (
    cod_grupo TEXT PRIMARY KEY,
    nome_grupo TEXT
);

-- modalidade_aplicacao definição
CREATE TABLE modalidade_aplicacao (
    cod_modalidade TEXT PRIMARY KEY,
    nome_modalidade TEXT
);

-- orgao definição
CREATE TABLE orgao (
    cod_orgao TEXT PRIMARY KEY,
    nome_orgao TEXT
);

-- poder definição
CREATE TABLE poder (
    cod_poder TEXT PRIMARY KEY,
    nome_poder TEXT
);

-- programa definição
CREATE TABLE programa (
    cod_programa TEXT PRIMARY KEY,
    nome_programa TEXT
);
```

```
-- subacao definição
CREATE TABLE subacao (
    cod_subacao TEXT PRIMARY KEY,
    nome_subacao TEXT
);

-- subelemento definição
CREATE TABLE subelemento (
    cod_subelemento TEXT PRIMARY KEY,
    nome_subelemento TEXT
);

-- subfuncao definição
CREATE TABLE subfuncao (
    cod_subfuncao TEXT PRIMARY KEY,
    nome_subfuncao TEXT
);

-- tempo definição
CREATE TABLE tempo (
    id INTEGER PRIMARY KEY,
    ano INTEGER,
    nro_mes INTEGER,
    mes TEXT,
    nro_bimestre INTEGER,
    bimestre TEXT,
    nro_trimestre INTEGER,
    trimestre TEXT,
    nro_quadrimestre INTEGER,
    quadrimestre TEXT,
    nro_semestre INTEGER,
    semestre TEXT
);

-- tipo_entidade definição
CREATE TABLE tipo_entidade (
    cod_tipo_entidade TEXT PRIMARY KEY,
    nome_tipo_entidade TEXT
);
```

```
-- tipo_fonte definição
CREATE TABLE tipo_fonte (
    cod_tipo TEXT PRIMARY KEY,
    nome_tipo TEXT
);

-- unidade_gestora definição
CREATE TABLE unidade_gestora (
    cod_ug TEXT PRIMARY KEY,
    nome_ug TEXT
);

-- uso definição
CREATE TABLE uso (
    cod_uso TEXT PRIMARY KEY,
    nome_uso TEXT
);

-- fato_despesa definição
CREATE TABLE fato_despesa (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        tempo_id INTEGER,
        cod_poder TEXT,
        cod_orgao TEXT,
        cod_ug TEXT,
        cod_gestao TEXT,
        cod_tipo_entidade TEXT,
        cod_funcao TEXT,
        cod_subfuncao TEXT,
        cod_programa TEXT,
        cod_acao TEXT,
        cod_subacao TEXT,
        cod_uso TEXT,
        cod_fonte TEXT,
        cod_grupo TEXT,
        cod_especificacao TEXT,
        cod_tipo TEXT,
        cod_categoria TEXT,
        cod_grupo_despesa TEXT,
        cod_modalidade TEXT,
```

```
    cod_elemento TEXT,
    cod_subelemento TEXT,
    cod_credor TEXT,
    indicador_emergencial TEXT,
    vl_dotacao_inicial REAL,
    vl_dotacao_atualizada REAL,
    vl_empenhado REAL,
    vl_liquidado REAL,
    vl_pago_orcamentario REAL,
    descricao_despesa_emergencial TEXT,
    CONSTRAINT FK_fato_despesa_acao FOREIGN KEY (cod_acao) REFERENCES
acao(cod_acao),
    CONSTRAINT FK_fato_despesa_categoria_economica_2 FOREIGN KEY
(cod_categoria) REFERENCES categoria_economica(cod_categoria),
    CONSTRAINT FK_fato_despesa_elemento_5 FOREIGN KEY (cod_elemento)
REFERENCES elemento(cod_elemento),
    CONSTRAINT FK_fato_despesa_especificacao_fonte_6 FOREIGN KEY
(cod_especificacao) REFERENCES especificacao_fonte(cod_especificacao),
    CONSTRAINT FK_fato_despesa_fonte_recurso_7 FOREIGN KEY (cod_fonte)
REFERENCES fonte_recurso(cod_fonte),
    CONSTRAINT FK_fato_despesa_funcao_8 FOREIGN KEY (cod_funcao)
REFERENCES funcao(cod_funcao),
    CONSTRAINT FK_fato_despesa_gestao_9 FOREIGN KEY (cod_gestao)
REFERENCES gestao(cod_gestao),
    CONSTRAINT FK_fato_despesa_grupo_despesa_10 FOREIGN KEY
(cod_grupo_despesa) REFERENCES grupo_despesa(cod_grupo),
    CONSTRAINT FK_fato_despesa_grupo_fonte_11 FOREIGN KEY (cod_grupo)
REFERENCES grupo_fonte(cod_grupo),
    CONSTRAINT FK_fato_despesa_modalidade_aplicacao_12 FOREIGN KEY
(cod_modalidade) REFERENCES modalidade_aplicacao(cod_modalidade),
    CONSTRAINT FK_fato_despesa_orgao_13 FOREIGN KEY (cod_orgao)
REFERENCES orgao(cod_orgao),
    CONSTRAINT FK_fato_despesa_programa_15 FOREIGN KEY (cod_programa)
REFERENCES programa(cod_programa),
    CONSTRAINT FK_fato_despesa_subacao_16 FOREIGN KEY (cod_subacao)
REFERENCES subacao(cod_subacao),
    CONSTRAINT FK_fato_despesa_subelemento_17 FOREIGN KEY
(cod_subelemento) REFERENCES subelemento(cod_subelemento),
    CONSTRAINT FK_fato_despesa_subfuncao_18 FOREIGN KEY (cod_subfuncao)
REFERENCES subfuncao(cod_subfuncao),
```

```
        CONSTRAINT FK_fato_despesa_tempo_19 FOREIGN KEY (tempo_id)
REFERENCES tempo(id),
        CONSTRAINT FK_fato_despesa_tipo_entidade_20 FOREIGN KEY
(cod_tipo_entidade) REFERENCES tipo_entidade(cod_tipo_entidade),
        CONSTRAINT FK_fato_despesa_tipo_fonte_21 FOREIGN KEY (cod_tipo)
REFERENCES tipo_fonte(cod_tipo),
        CONSTRAINT FK_fato_despesa_unidade_gestora_22 FOREIGN KEY (cod_ug)
REFERENCES unidade_gestora(cod_ug),
        CONSTRAINT FK_fato_despesa_uso_23 FOREIGN KEY (cod_uso) REFERENCES
uso(cod_uso)
    );
```

APENDICE E – DDL STAR SCHEMA EM INGLÊS

```
-- action definition
CREATE TABLE action (
    action_code TEXT PRIMARY KEY,
    action_name TEXT
);

-- economic_category definition
CREATE TABLE economic_category (
    category_code TEXT PRIMARY KEY,
    category_name TEXT
);

-- element definition
CREATE TABLE element (
    element_code TEXT PRIMARY KEY,
    element_name TEXT
);

-- funding_specification definition
CREATE TABLE funding_specification (
    specification_code TEXT PRIMARY KEY,
    specification_name TEXT
);

-- funding_source definition
CREATE TABLE funding_source (
    source_code TEXT PRIMARY KEY,
    source_name TEXT
);

-- function definition
CREATE TABLE function (
    function_code TEXT PRIMARY KEY,
    function_name TEXT
);

-- management definition
CREATE TABLE management (
```

```
    management_code TEXT PRIMARY KEY,
    management_name TEXT
);

-- expense_group definition
CREATE TABLE expense_group (
    group_code TEXT PRIMARY KEY,
    group_name TEXT
);

-- source_group definition
CREATE TABLE source_group (
    group_code TEXT PRIMARY KEY,
    group_name TEXT
);

-- application_modality definition
CREATE TABLE application_modality (
    modality_code TEXT PRIMARY KEY,
    modality_name TEXT
);

-- agency definition
CREATE TABLE agency (
    agency_code TEXT PRIMARY KEY,
    agency_name TEXT
);

-- power_branch definition
CREATE TABLE power_branch (
    power_code TEXT PRIMARY KEY,
    power_name TEXT
);

-- program definition
CREATE TABLE program (
    program_code TEXT PRIMARY KEY,
    program_name TEXT
);
```

```
-- subaction definition
CREATE TABLE subaction (
    subaction_code TEXT PRIMARY KEY,
    subaction_name TEXT
);

-- subelement definition
CREATE TABLE subelement (
    subelement_code TEXT PRIMARY KEY,
    subelement_name TEXT
);

-- subfunction definition
CREATE TABLE subfunction (
    subfunction_code TEXT PRIMARY KEY,
    subfunction_name TEXT
);

-- time definition
CREATE TABLE time (
    id INTEGER PRIMARY KEY,
    year INTEGER,
    month_number INTEGER,
    month TEXT,
    bimester_number INTEGER,
    bimester TEXT,
    trimester_number INTEGER,
    trimester TEXT,
    quadrimester_number INTEGER,
    quadrimester TEXT,
    semester_number INTEGER,
    semester TEXT
);

-- entity_type definition
CREATE TABLE entity_type (
    entity_type_code TEXT PRIMARY KEY,
    entity_type_name TEXT
);
```

```
-- source_type definition
CREATE TABLE source_type (
    type_code TEXT PRIMARY KEY,
    type_name TEXT
);

-- management_unit definition
CREATE TABLE management_unit (
    unit_code TEXT PRIMARY KEY,
    unit_name TEXT
);

-- usage definition
CREATE TABLE usage (
    usage_code TEXT PRIMARY KEY,
    usage_name TEXT
);

-- expense_fact definition
CREATE TABLE expense_fact (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    time_id INTEGER,
    power_code TEXT,
    agency_code TEXT,
    unit_code TEXT,
    management_code TEXT,
    entity_type_code TEXT,
    function_code TEXT,
    subfunction_code TEXT,
    program_code TEXT,
    action_code TEXT,
    subaction_code TEXT,
    usage_code TEXT,
    source_code TEXT,
    group_code TEXT,
    specification_code TEXT,
    type_code TEXT,
    category_code TEXT,
    expense_group_code TEXT,
    modality_code TEXT,
```

```
    element_code TEXT,
    subelement_code TEXT,
    creditor_code TEXT,
    emergency_indicator TEXT,
    initial_budget_value REAL,
    updated_budget_value REAL,
    committed_value REAL,
    settled_value REAL,
    paid_budget_value REAL,
    emergency_expense_description TEXT,
    CONSTRAINT FK_expense_fact_action FOREIGN KEY (action_code) REFERENCES
action(action_code),
    CONSTRAINT FK_expense_fact_economic_category FOREIGN KEY (category_code)
REFERENCES economic_category(category_code),
    CONSTRAINT FK_expense_fact_element FOREIGN KEY (element_code) REFERENCES
element(element_code),
    CONSTRAINT FK_expense_fact_funding_specification FOREIGN KEY (specification_code)
REFERENCES funding_specification(specification_code),
    CONSTRAINT FK_expense_fact_funding_source FOREIGN KEY (source_code) REFERENCES
funding_source(source_code),
    CONSTRAINT FK_expense_fact_function FOREIGN KEY (function_code) REFERENCES
function(function_code),
    CONSTRAINT FK_expense_fact_management FOREIGN KEY (management_code)
REFERENCES management(management_code),
    CONSTRAINT FK_expense_fact_expense_group FOREIGN KEY (expense_group_code)
REFERENCES expense_group(group_code),
    CONSTRAINT FK_expense_fact_source_group FOREIGN KEY (group_code) REFERENCES
source_group(group_code),
    CONSTRAINT FK_expense_fact_application_modality FOREIGN KEY (modality_code)
REFERENCES application_modality(modality_code),
    CONSTRAINT FK_expense_fact_agency FOREIGN KEY (agency_code) REFERENCES
agency(agency_code),
    CONSTRAINT FK_expense_fact_program FOREIGN KEY (program_code) REFERENCES
program(program_code),
    CONSTRAINT FK_expense_fact_subaction FOREIGN KEY (subaction_code) REFERENCES
subaction(subaction_code),
    CONSTRAINT FK_expense_fact_subelement FOREIGN KEY (subelement_code) REFERENCES
subelement(subelement_code),
    CONSTRAINT FK_expense_fact_subfunction FOREIGN KEY (subfunction_code) REFERENCES
subfunction(subfunction_code),
```

```
    CONSTRAINT FK_expense_fact_time FOREIGN KEY (time_id) REFERENCES time(id),
    CONSTRAINT FK_expense_fact_entity_type FOREIGN KEY (entity_type_code) REFERENCES
entity_type(entity_type_code),
    CONSTRAINT FK_expense_fact_source_type FOREIGN KEY (type_code) REFERENCES
source_type(type_code),
    CONSTRAINT FK_expense_fact_management_unit FOREIGN KEY (unit_code) REFERENCES
management_unit(unit_code),
    CONSTRAINT FK_expense_fact_usage FOREIGN KEY (usage_code) REFERENCES
usage(usage_code)
);
```

APENDICE F – PROMPT TABELA ÚNICA EM PORTUGUÊS

Você é um gerador de SQL para SQLite.

Não tente calcular valores, apenas gerar um comando SQL

*Dada a pergunta e o esquema do banco de dados abaixo, gere **apenas UM comando SQL válido**, sem explicações ou texto extra.*

O comando deve terminar com ponto e vírgula `;`.

Não escreva nada além do SQL.

Esquema do banco de dados:

CREATE TABLE expense_fact (year TEXT, month_number TEXT, month TEXT, bimester_number TEXT, bimester TEXT, trimester_number TEXT, trimester TEXT, quadrimester_number TEXT, quadrimester TEXT, semester_number TEXT, semester TEXT, power_code TEXT, power_name TEXT, agency_code TEXT, agency_name TEXT, unit_code TEXT, management_unit TEXT, management_code TEXT, management_name TEXT, entity_type_code TEXT, entity_type TEXT, function_code TEXT, function_name TEXT, subfunction_code TEXT, subfunction_name TEXT, program_code TEXT, program_name TEXT, action_code TEXT, action_name TEXT, subaction_code TEXT, subaction_name TEXT, usage_code TEXT, usage_name TEXT, source_code TEXT, source_name TEXT, group_code TEXT, group_name TEXT, specification_code TEXT, specification_name TEXT, type_code TEXT, type_name TEXT, category_code TEXT, category_name TEXT, expense_group_code TEXT, expense_group_name TEXT, modality_code TEXT, modality_name TEXT, element_code TEXT, element_name TEXT, subelement_code TEXT, subelement_name TEXT, creditor_code TEXT, creditor_name TEXT, emergency_indicator TEXT, emergency_description TEXT, initial_budget_value REAL, updated_budget_value REAL, committed_value REAL, settled_value REAL, paid_budget_value REAL);

Escreva um comando sql valido para a pergunta com base no schema:

Quais são os anos disponíveis nos dados?

SQL: ###Resposta: SELECT DISTINCT year FROM expense_fact;

APENDICE G – PROMPT TABELA ÚNICA EM INGLÊS

You are an SQL generator for SQLite.

*Given the question and the database schema below, generate only **ONE** valid SQL command,with no extra explanations or text.*

The command must end with a semicolon ';'.

Do not write anything other than the SQL.

Database schema: CREATE TABLE expense_fact (year TEXT, month_number TEXT, month TEXT, bimester_number TEXT, bimester TEXT, trimester_number TEXT, trimester TEXT, quadrimester_number TEXT, quadrimester TEXT, semester_number TEXT, semester TEXT, power_code TEXT, power_name TEXT, agency_code TEXT, agency_name TEXT, unit_code TEXT, management_unit TEXT, management_code TEXT, management_name TEXT, entity_type_code TEXT, entity_type TEXT, function_code TEXT, function_name TEXT, subfunction_code TEXT, subfunction_name TEXT, program_code TEXT, program_name TEXT, action_code TEXT, action_name TEXT, subaction_code TEXT, subaction_name TEXT, usage_code TEXT, usage_name TEXT, source_code TEXT, source_name TEXT, group_code TEXT, group_name TEXT, specification_code TEXT, specification_name TEXT, type_code TEXT, type_name TEXT, category_code TEXT, category_name TEXT, expense_group_code TEXT, expense_group_name TEXT, modality_code TEXT, modality_name TEXT, element_code TEXT, element_name TEXT, subelement_code TEXT, subelement_name TEXT, creditor_code TEXT, creditor_name TEXT, emergency_indicator TEXT, emergency_description TEXT, initial_budget_value REAL, updated_budget_value REAL, committed_value REAL, settled_value REAL, paid_budget_value REAL);

Write an SQL command for the question: Which years are available in the data?

SQL: ###Response: SELECT DISTINCT year FROM expense_fact;

APENDICE H – PROMPT STAR SCHEMA EM PORTUGUÊS

Você é um gerador de SQL para SQLite.

Não tente calcular valores, apenas gerar um comando SQL

*Dada a pergunta e o esquema do banco de dados abaixo, gere **apenas UM comando SQL válido**, sem explicações ou texto extra.*

O comando deve terminar com ponto e vírgula `;`.

Não escreva nada além do SQL.

Esquema do banco de dados:

CREATE TABLE action (action_code TEXT PRIMARY KEY, action_name TEXT); CREATE TABLE economic_category (category_code TEXT PRIMARY KEY, category_name TEXT); CREATE TABLE element (element_code TEXT PRIMARY KEY, element_name TEXT); CREATE TABLE funding_specification (specification_code TEXT PRIMARY KEY, specification_name TEXT); CREATE TABLE funding_source (source_code TEXT PRIMARY KEY, source_name TEXT); CREATE TABLE function (function_code TEXT PRIMARY KEY, function_name TEXT); CREATE TABLE management (management_code TEXT PRIMARY KEY, management_name TEXT); CREATE TABLE expense_group (expense_group_code TEXT PRIMARY KEY, expense_group_name TEXT); CREATE TABLE source_group (group_code TEXT PRIMARY KEY, group_name TEXT); CREATE TABLE application_modality (modality_code TEXT PRIMARY KEY, modality_name TEXT); CREATE TABLE agency (agency_code TEXT PRIMARY KEY, agency_name TEXT); CREATE TABLE power_branch (power_code TEXT PRIMARY KEY, power_name TEXT); CREATE TABLE program (program_code TEXT PRIMARY KEY, program_name TEXT); CREATE TABLE subaction (subaction_code TEXT PRIMARY KEY, subaction_name TEXT); CREATE TABLE subelement (subelement_code TEXT PRIMARY KEY, subelement_name TEXT); CREATE TABLE subfunction (subfunction_code TEXT PRIMARY KEY, subfunction_name TEXT); CREATE TABLE time (id INTEGER PRIMARY KEY, year INTEGER, month_number INTEGER, month TEXT, bimester_number INTEGER, bimester TEXT, trimester_number INTEGER, trimester TEXT, quadrimester_number INTEGER, quadrimester TEXT, semester_number INTEGER, semester TEXT); CREATE TABLE entity_type (entity_type_code TEXT PRIMARY KEY, entity_type_name TEXT); CREATE TABLE source_type (type_code TEXT PRIMARY KEY, type_name TEXT); CREATE TABLE management_unit (unit_code TEXT PRIMARY KEY, unit_name TEXT); CREATE TABLE usage (usage_code TEXT PRIMARY KEY, usage_name TEXT); CREATE TABLE expense_fact (id INTEGER PRIMARY KEY AUTOINCREMENT, time_id INTEGER, power_code TEXT, agency_code TEXT, unit_code TEXT, management_code TEXT, entity_type_code TEXT, function_code TEXT, subfunction_code TEXT, program_code TEXT, action_code TEXT, subaction_code TEXT, usage_code TEXT, source_code TEXT, group_code TEXT, specification_code TEXT, type_code TEXT, category_code TEXT, expense_group_code TEXT, modality_code TEXT, element_code TEXT, subelement_code TEXT, creditor_code TEXT, emergency_indicator TEXT, initial_budget_value REAL, updated_budget_value REAL, committed_value REAL, settled_value REAL, paid_budget_value REAL,

emergency_expense_description TEXT, CONSTRAINT FK_expense_fact_action FOREIGN KEY (action_code) REFERENCES action(action_code), CONSTRAINT FK_expense_fact_economic_category FOREIGN KEY (category_code) REFERENCES economic_category(category_code), CONSTRAINT FK_expense_fact_element FOREIGN KEY (element_code) REFERENCES element(element_code), CONSTRAINT FK_expense_fact_funding_specification FOREIGN KEY (specification_code) REFERENCES funding_specification(specification_code), CONSTRAINT FK_expense_fact_funding_source FOREIGN KEY (source_code) REFERENCES funding_source(source_code), CONSTRAINT FK_expense_fact_function FOREIGN KEY (function_code) REFERENCES function(function_code), CONSTRAINT FK_expense_fact_management FOREIGN KEY (management_code) REFERENCES management(management_code), CONSTRAINT FK_expense_fact_expense_group FOREIGN KEY (expense_group_code) REFERENCES expense_group(expense_group_code), CONSTRAINT FK_expense_fact_source_group FOREIGN KEY (group_code) REFERENCES source_group(group_code), CONSTRAINT FK_expense_fact_application_modality FOREIGN KEY (modality_code) REFERENCES application_modality(modality_code), CONSTRAINT FK_expense_fact_agency FOREIGN KEY (agency_code) REFERENCES agency(agency_code), CONSTRAINT FK_expense_fact_program FOREIGN KEY (program_code) REFERENCES program(program_code), CONSTRAINT FK_expense_fact_subaction FOREIGN KEY (subaction_code) REFERENCES subaction(subaction_code), CONSTRAINT FK_expense_fact_subelement FOREIGN KEY (subelement_code) REFERENCES subelement(subelement_code), CONSTRAINT FK_expense_fact_subfunction FOREIGN KEY (subfunction_code) REFERENCES subfunction(subfunction_code), CONSTRAINT FK_expense_fact_time FOREIGN KEY (time_id) REFERENCES time(id), CONSTRAINT FK_expense_fact_entity_type FOREIGN KEY (entity_type_code) REFERENCES entity_type(entity_type_code), CONSTRAINT FK_expense_fact_source_type FOREIGN KEY (type_code) REFERENCES source_type(type_code), CONSTRAINT FK_expense_fact_management_unit FOREIGN KEY (unit_code) REFERENCES management_unit(unit_code), CONSTRAINT FK_expense_fact_usage FOREIGN KEY (usage_code) REFERENCES usage(usage_code)); CREATE TABLE sqlite_sequence(name,seq);

Escreva um comando sql valido para a pergunta com base no schema:

Quais são os anos disponíveis nos dados?

SQL: ###Resposta: SELECT DISTINCT year FROM time;

APENDICE I – PROMPT STAR SCHEMA EM INGLÊS

You are an SQL generator for SQLite.

*Given the question and the database schema below, generate only **ONE** valid SQL command,with no extra explanations or text.*

The command must end with a semicolon ';'.

Do not write anything other than the SQL.Database

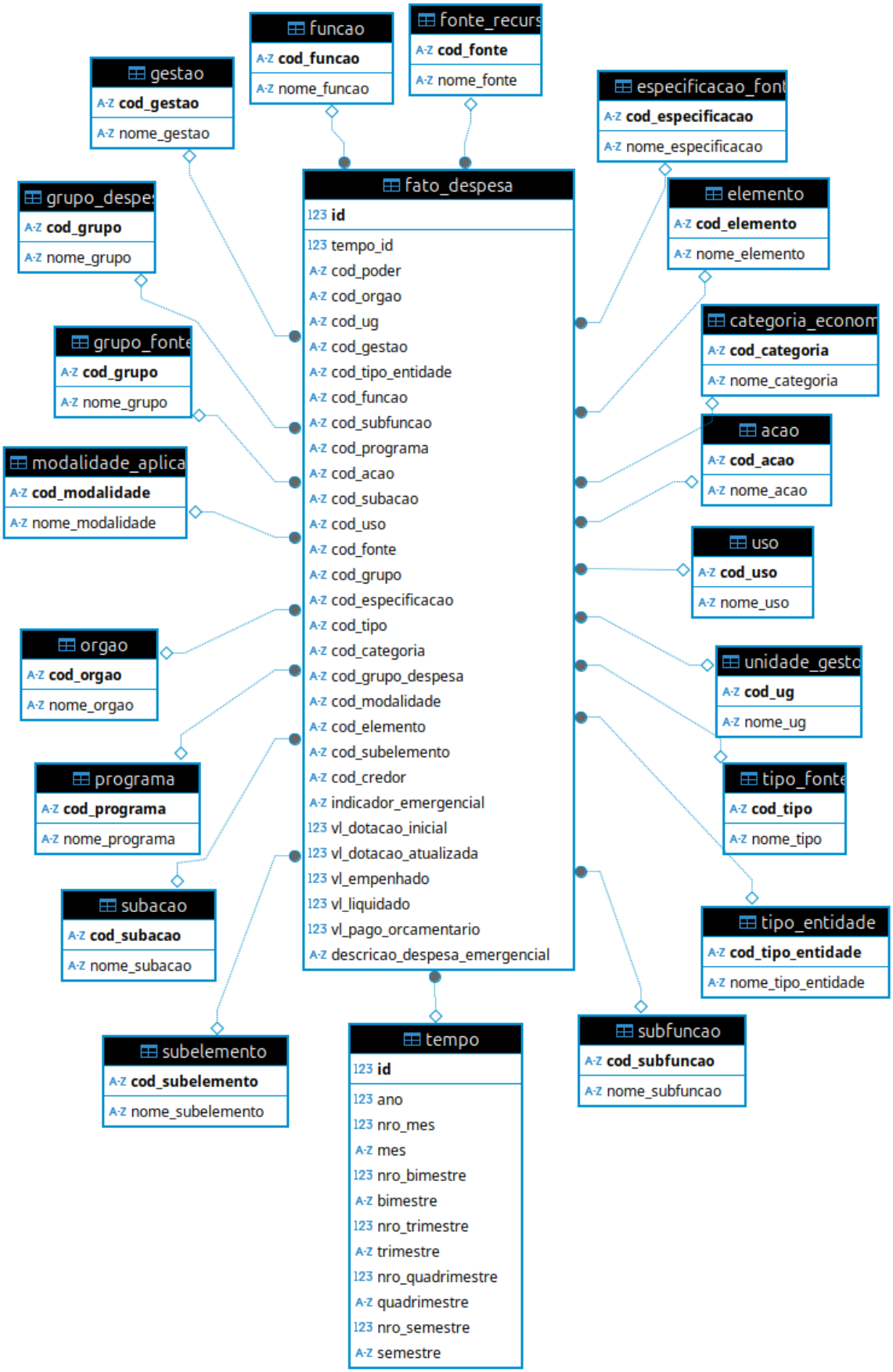
schema: CREATE TABLE action (action_code TEXT PRIMARY KEY, action_name TEXT); CREATE TABLE economic_category (category_code TEXT PRIMARY KEY, category_name TEXT); CREATE TABLE element (element_code TEXT PRIMARY KEY, element_name TEXT); CREATE TABLE funding_specification (specification_code TEXT PRIMARY KEY, specification_name TEXT); CREATE TABLE funding_source (source_code TEXT PRIMARY KEY, source_name TEXT); CREATE TABLE function (function_code TEXT PRIMARY KEY, function_name TEXT); CREATE TABLE management (management_code TEXT PRIMARY KEY, management_name TEXT); CREATE TABLE expense_group (expense_group_code TEXT PRIMARY KEY, expense_group_name TEXT); CREATE TABLE source_group (group_code TEXT PRIMARY KEY, group_name TEXT); CREATE TABLE application_modality (modality_code TEXT PRIMARY KEY, modality_name TEXT); CREATE TABLE agency (agency_code TEXT PRIMARY KEY, agency_name TEXT); CREATE TABLE power_branch (power_code TEXT PRIMARY KEY, power_name TEXT); CREATE TABLE program (program_code TEXT PRIMARY KEY, program_name TEXT); CREATE TABLE subaction (subaction_code TEXT PRIMARY KEY, subaction_name TEXT); CREATE TABLE subelement (subelement_code TEXT PRIMARY KEY, subelement_name TEXT); CREATE TABLE subfunction (subfunction_code TEXT PRIMARY KEY, subfunction_name TEXT); CREATE TABLE time (id INTEGER PRIMARY KEY, year INTEGER, month_number INTEGER, month TEXT, bimester_number INTEGER, bimester TEXT, trimester_number INTEGER, trimester TEXT, quadrimester_number INTEGER, quadrimester TEXT, semester_number INTEGER, semester TEXT); CREATE TABLE entity_type (entity_type_code TEXT PRIMARY KEY, entity_type_name TEXT); CREATE TABLE source_type (type_code TEXT PRIMARY KEY, type_name TEXT); CREATE TABLE management_unit (unit_code TEXT PRIMARY KEY, unit_name TEXT); CREATE TABLE usage (usage_code TEXT PRIMARY KEY, usage_name TEXT); CREATE TABLE expense_fact (id INTEGER PRIMARY KEY AUTOINCREMENT, time_id INTEGER, power_code TEXT, agency_code TEXT, unit_code TEXT, management_code TEXT, entity_type_code TEXT, function_code TEXT, subfunction_code TEXT, program_code TEXT, action_code TEXT, subaction_code TEXT, usage_code TEXT, source_code TEXT, group_code TEXT, specification_code TEXT, type_code TEXT, category_code TEXT, expense_group_code TEXT, modality_code TEXT, element_code TEXT, subelement_code TEXT, creditor_code TEXT, emergency_indicator TEXT, initial_budget_value REAL, updated_budget_value REAL, committed_value REAL, settled_value REAL, paid_budget_value REAL, emergency_expense_description TEXT, CONSTRAINT FK_expense_fact_action FOREIGN KEY (action_code) REFERENCES action(action_code), CONSTRAINT

FK_expense_fact_economic_category FOREIGN KEY (category_code) REFERENCES economic_category(category_code), CONSTRAINT FK_expense_fact_element FOREIGN KEY (element_code) REFERENCES element(element_code), CONSTRAINT FK_expense_fact_funding_specification FOREIGN KEY (specification_code) REFERENCES funding_specification(specification_code), CONSTRAINT FK_expense_fact_funding_source FOREIGN KEY (source_code) REFERENCES funding_source(source_code), CONSTRAINT FK_expense_fact_function FOREIGN KEY (function_code) REFERENCES function(function_code), CONSTRAINT FK_expense_fact_management FOREIGN KEY (management_code) REFERENCES management(management_code), CONSTRAINT FK_expense_fact_expense_group FOREIGN KEY (expense_group_code) REFERENCES expense_group(expense_group_code), CONSTRAINT FK_expense_fact_source_group FOREIGN KEY (group_code) REFERENCES source_group(group_code), CONSTRAINT FK_expense_fact_application_modality FOREIGN KEY (modality_code) REFERENCES application_modality(modality_code), CONSTRAINT FK_expense_fact_agency FOREIGN KEY (agency_code) REFERENCES agency(agency_code), CONSTRAINT FK_expense_fact_program FOREIGN KEY (program_code) REFERENCES program(program_code), CONSTRAINT FK_expense_fact_subaction FOREIGN KEY (subaction_code) REFERENCES subaction(subaction_code), CONSTRAINT FK_expense_fact_subelement FOREIGN KEY (subelement_code) REFERENCES subelement(subelement_code), CONSTRAINT FK_expense_fact_subfunction FOREIGN KEY (subfunction_code) REFERENCES subfunction(subfunction_code), CONSTRAINT FK_expense_fact_time FOREIGN KEY (time_id) REFERENCES time(id), CONSTRAINT FK_expense_fact_entity_type FOREIGN KEY (entity_type_code) REFERENCES entity_type(entity_type_code), CONSTRAINT FK_expense_fact_source_type FOREIGN KEY (type_code) REFERENCES source_type(type_code), CONSTRAINT FK_expense_fact_management_unit FOREIGN KEY (unit_code) REFERENCES management_unit(unit_code), CONSTRAINT FK_expense_fact_usage FOREIGN KEY (usage_code) REFERENCES usage(usage_code)); CREATE TABLE sqlite_sequence(name,seq);

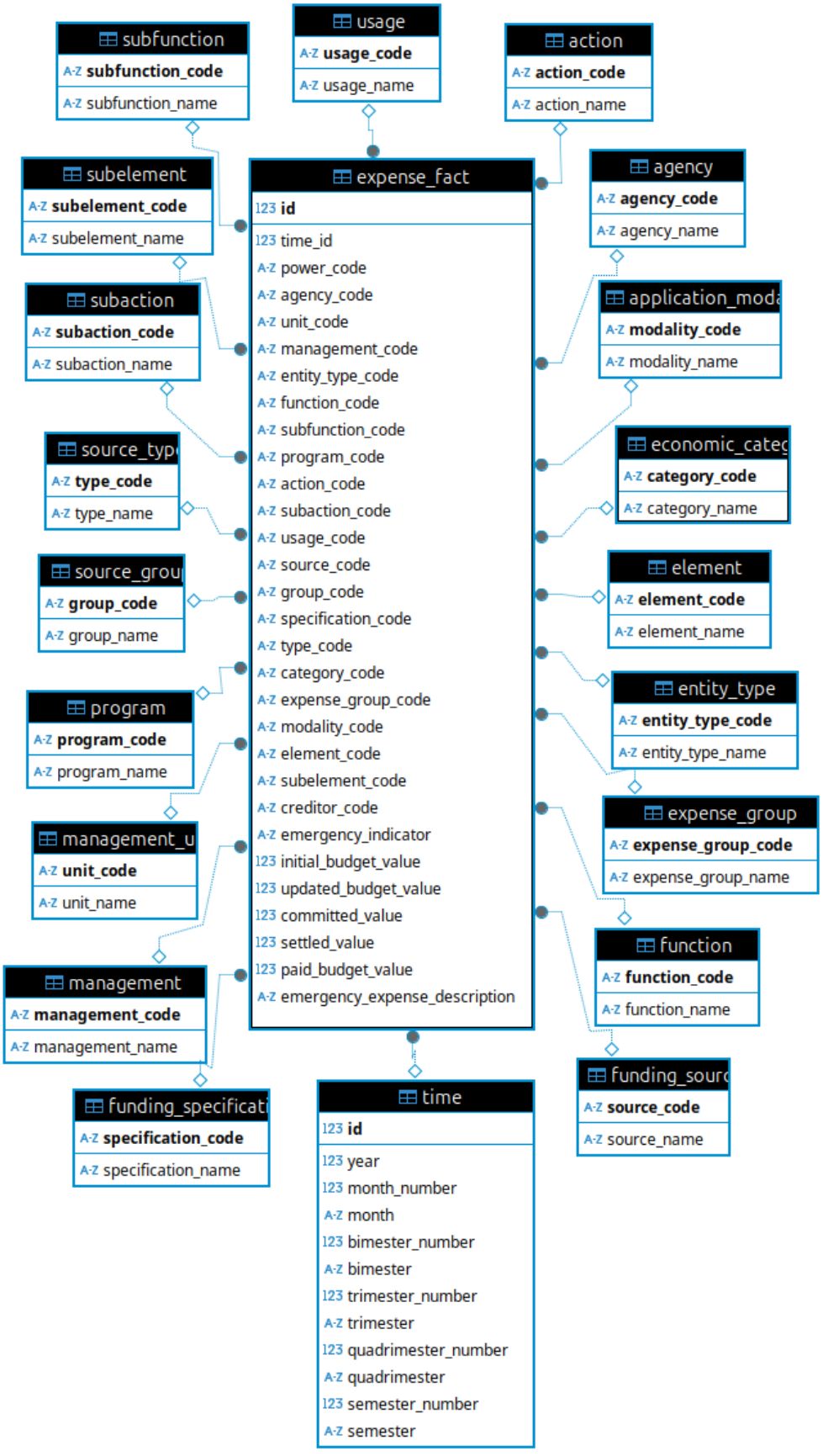
Write an SQL command for the question: Which years are available in the data?

SQL: ###Response: SELECT DISTINCT year FROM time;

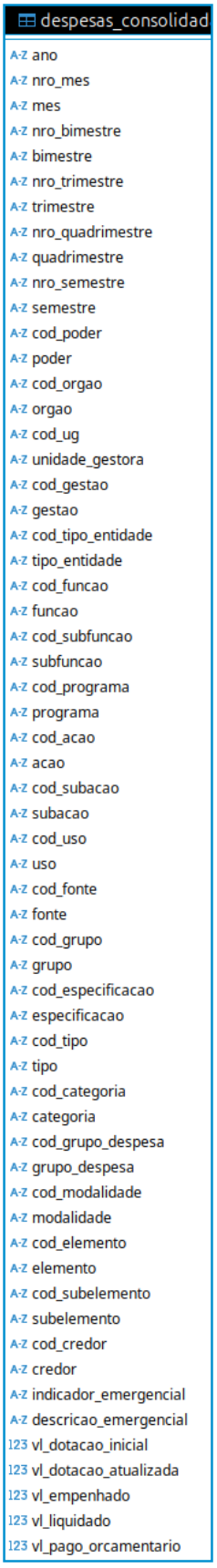
APENDICE J – BASE DE DADOS EM STAR SCHEMA EM PORTUGUÊS



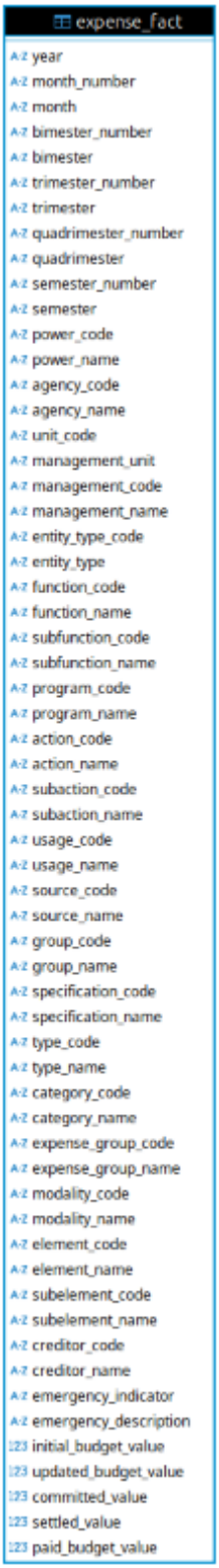
APENDICE K – BASE DE DADOS EM STAR SCHEMA EM INGLÊS



APENDICE L – BASE DE DADOS EM TABELA ÚNICA EM PORTUGUÊS



APENDICE M – BASE DE DADOS EM TABELA ÚNICA EM INGLÊS



UDESC UNIVERSIDADE DO ESTADO DE SANTA CATARINA | JOINVILLE CENTRO DE CIÊNCIAS TECNOLÓGICAS

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
BIBLIOTECA UNIVERSITÁRIA
REPOSITÓRIO INSTITUCIONAL

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

ATESTADO DE VERSÃO FINAL

Eu, André Tavares da Silva, professor(a) do curso de Mestrado em Computação Aplicada, declaro que esta é a versão final aprovada pela comissão julgadora da dissertação/tese intitulada: **"Comparativo do Impacto do Idioma e da Modelagem de Dados no Desempenho de Modelos de Linguagem na Tarefa de Text-to-SQL"** em 28/11/2015 de autoria do acadêmico Jonas Orso.

Joinville, 14 de Janeiro de 2026.

Assinatura digital do(a) orientador(a):

_____

André Tavares da Silva

Assinaturas do documento

Código para verificação: **P8RX070M**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

ANDRE TAVARES DA SILVA (CPF: 908.XXX.020-XX) em 14/01/2026 às 18:11:58
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:44:17 e válido até 30/03/2118 - 12:44:17.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwNTI5NDNfNTI5OTVfMjAyNF9QOFJYMDcwTQ== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00052943/2024** e o código **P8RX070M** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.